SQLBolt

SQL Lesson 1: SELECT queries 101

```
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
```

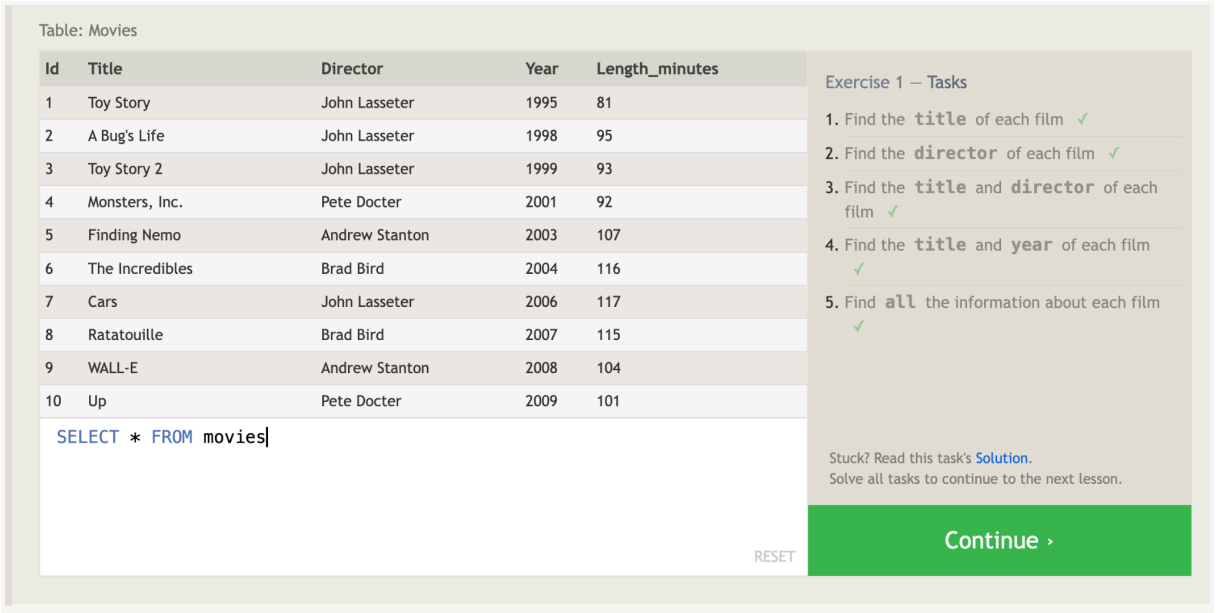


Table: Movies

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

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

```
1  SELECT * FROM movies where id = 6;
2  SELECT Title from movies WHERE year BETWEEN 2000 AND 2010;
    1.  3  SELECT Title from movies WHERE year NOT BETWEEN 2000 AND 2010;
4  SELECT * FROM movies Limit 5;
```

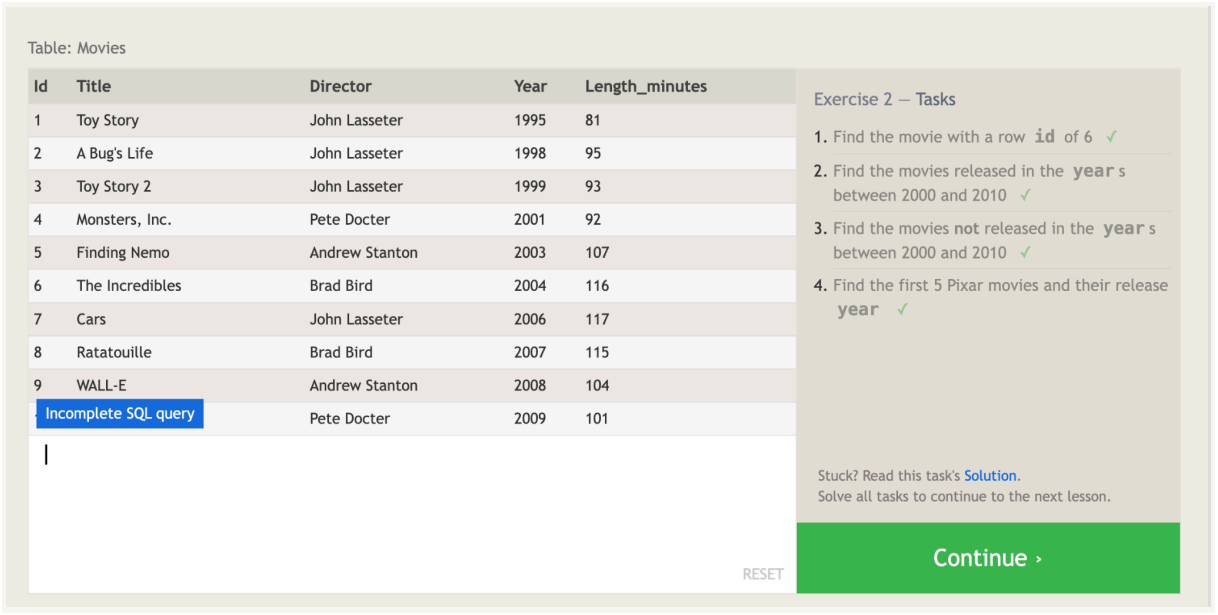


Table: Movies

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

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

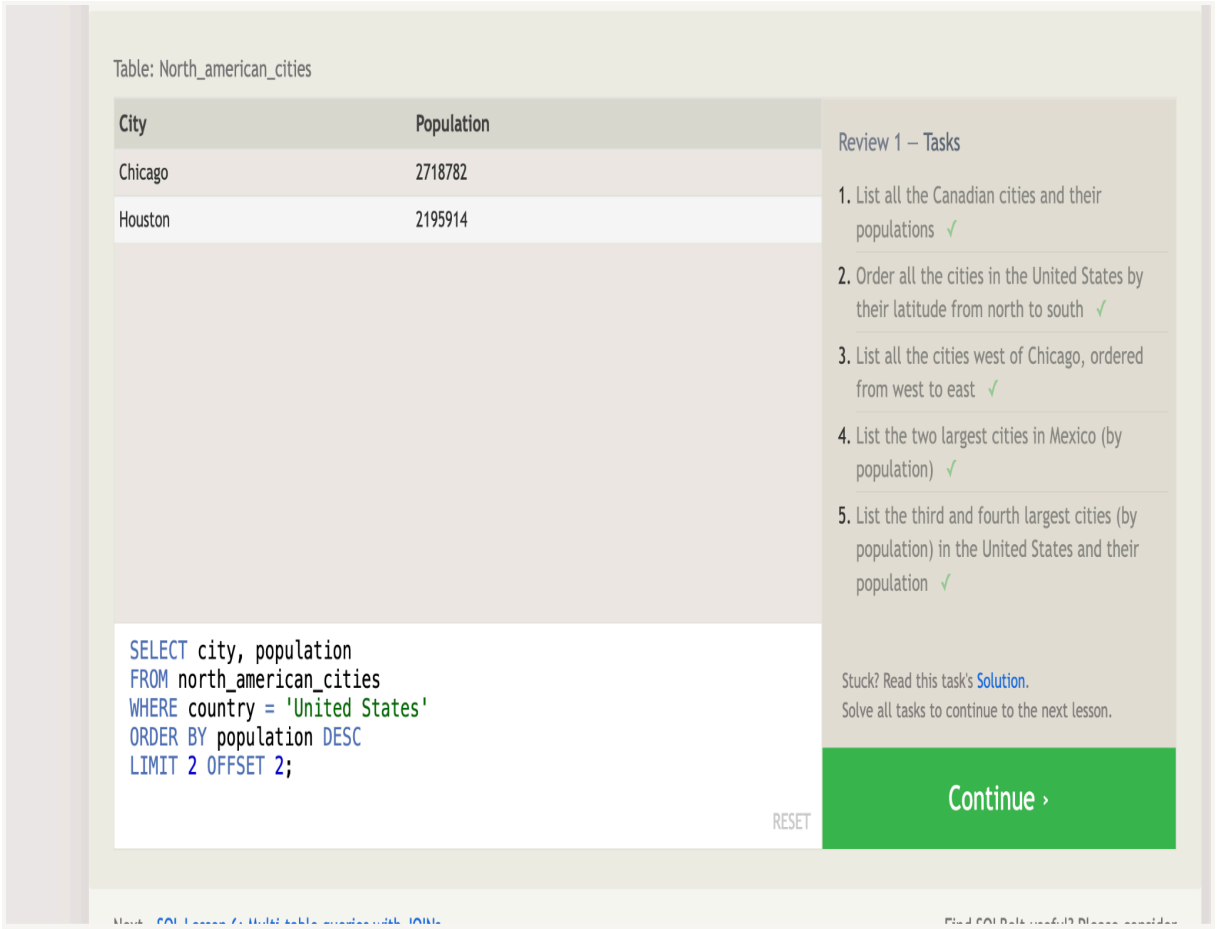
1. SELECT *FROM movies where Title Like 'Toy Story%';
2. SELECT *FROM movies where Director Like 'John Lasseter%';
3. SELECT * FROM movies where Director NOT like 'John Lasseter';
4. SELECT * FROM movies where Title like 'WALL-%';

SQL Lesson 4: Filtering and sorting Query results

1. SELECT DISTINCT director FROM movies ORDER BY director ASC;
2. SELECT *FROM movies ORDER BY year DESC LIMIT 4;
3. SELECT *FROM movies ORDER BY Title ASC LIMIT 5;
4. SELECT *FROM movies ORDER BY Title ASC LIMIT 5 OFFSET 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 < (SELECT longitude FROM north_american_cities WHERE city = 'Chicago') ORDER BY longitude ASC;
4. SELECT city, population FROM north_american_cities WHERE country = 'Mexico' ORDER BY population DESC LIMIT 2;
5. SELECT city, population FROM north_american_cities WHERE country = 'United States' ORDER BY population DESC LIMIT 2 OFFSET 2;



SQL Lesson 6: Multi-table queries with JOINs

1. SELECT m.title, b.domestic_sales, b.international_sales FROM movies m JOIN boxoffice b ON m.id = b.movie_id;
2. SELECT m.title, b.domestic_sales, b.international_sales FROM movies m JOIN boxoffice b ON m.id = b.movie_id WHERE b.international_sales > b.domestic_sales;
3. SELECT m.title, b.rating FROM movies m JOIN boxoffice b ON m.id = b.movie_id ORDER BY b.rating DESC;

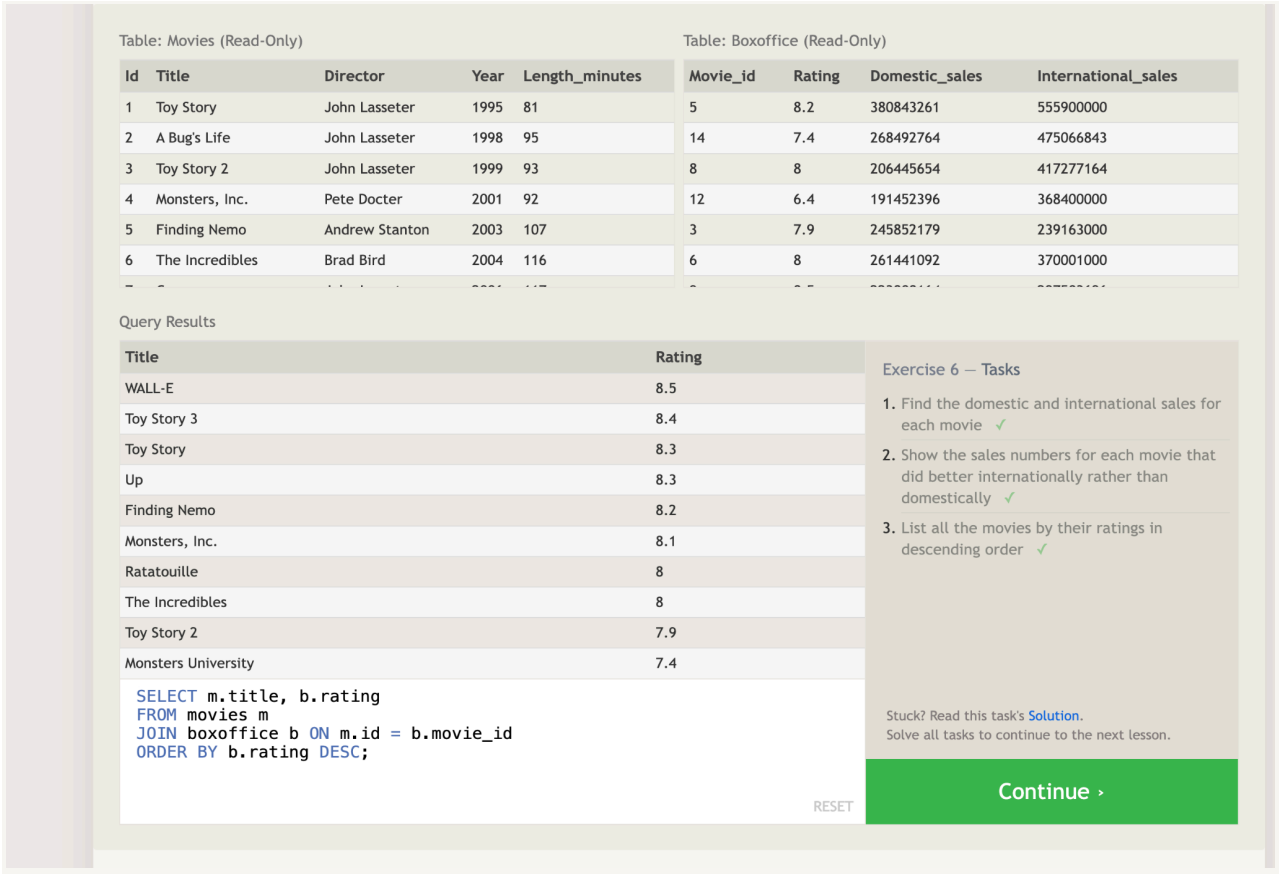


Table: Movies (Read-Only)

Id	Title	Director	Year	Length_minutes
1	Toy Story	John Lasseter	1995	81
2	A Bug's Life	John Lasseter	1998	95
3	Toy Story 2	John Lasseter	1999	93
4	Monsters, Inc.	Pete Docter	2001	92
5	Finding Nemo	Andrew Stanton	2003	107
6	The Incredibles	Brad Bird	2004	116

Table: Boxoffice (Read-Only)

Movie_id	Rating	Domestic_sales	International_sales
5	8.2	380843261	555900000
14	7.4	268492764	475066843
8	8	206445654	417277164
12	6.4	191452396	368400000
3	7.9	245852179	239163000
6	8	261441092	370001000

Query Results

Title	Rating
WALL-E	8.5
Toy Story 3	8.4
Toy Story	8.3
Up	8.3
Finding Nemo	8.2
Monsters, Inc.	8.1
Ratatouille	8
The Incredibles	8
Toy Story 2	7.9
Monsters University	7.4

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

RESET

Exercise 6 — Tasks

1. Find the domestic and international sales for each movie ✓
2. Show the sales numbers for each movie that did better internationally rather than domestically ✓
3. List all the movies by their ratings in descending order ✓

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

Continue ›

SQL Lesson 7: OUTER JOINs

1. SELECT DISTINCT building FROM employees;
2. SELECT building_name, capacity FROM buildings;

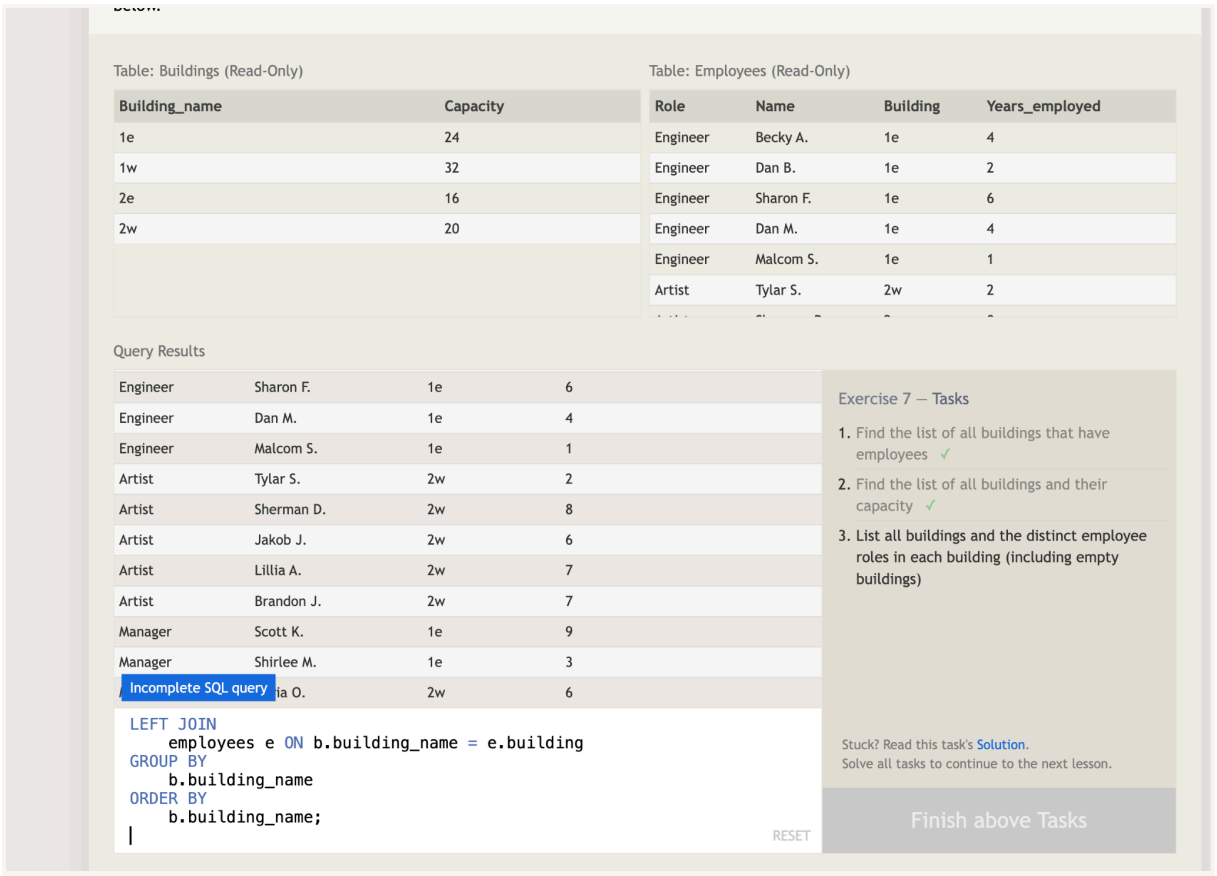


Table: Buildings (Read-Only)

Building_name	Capacity
1e	24
1w	32
2e	16
2w	20

Table: Employees (Read-Only)

Role	Name	Building	Years_employed
Engineer	Becky A.	1e	4
Engineer	Dan B.	1e	2
Engineer	Sharon F.	1e	6
Engineer	Dan M.	1e	4
Engineer	Malcom S.	1e	1
Artist	Tylar S.	2w	2

Query Results

Engineer	Sharon F.	1e	6
Engineer	Dan M.	1e	4
Engineer	Malcom S.	1e	1
Artist	Tylar S.	2w	2
Artist	Sherman D.	2w	8
Artist	Jakob J.	2w	6
Artist	Lillia A.	2w	7
Artist	Brandon J.	2w	7
Manager	Scott K.	1e	9
Manager	Shirlee M.	1e	3
	ia O.	2w	6

Incomplete SQL query

```
LEFT JOIN
    employees e ON b.building_name = e.building
GROUP BY
    b.building_name
ORDER BY
    b.building_name;
```

RESET

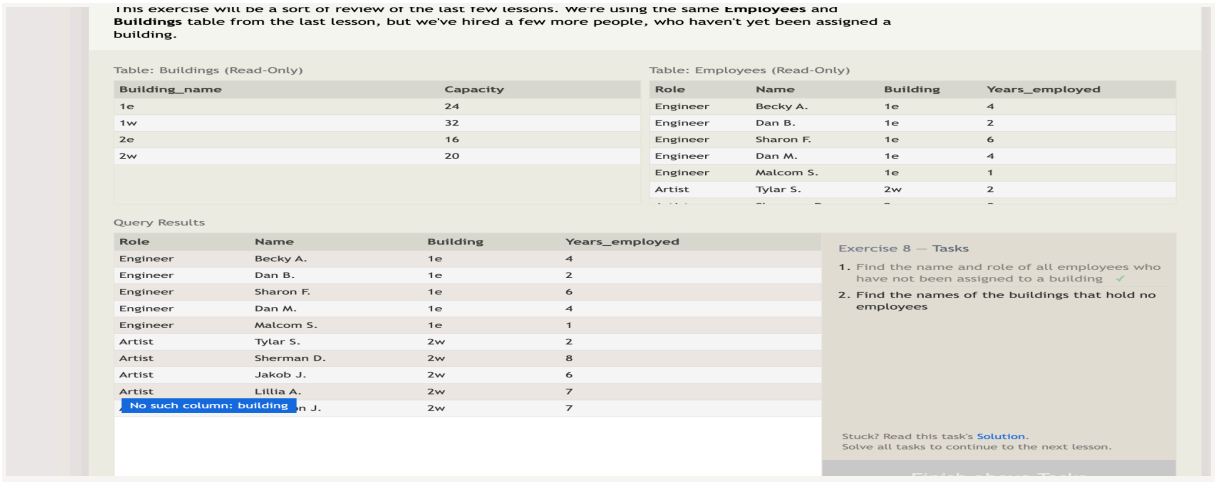
Exercise 7 – Tasks

1. Find the list of all buildings that have employees ✓
2. Find the list of all buildings and their capacity ✓
3. List all buildings and the distinct employee roles in each building (including empty buildings)

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

Finish above Tasks

SQL Lesson 8: A short note on NULLs

1. SELECT name, role FROM employees WHERE building IS NULL;
2. 

SQL Lesson 9: Queries with expressions

1. SELECT m.title, (b.domestic_sales + b.international_sales) / 1000000 AS combined_sales_millions FROM movies m JOIN boxoffice b ON m.id = b.movie_id;
2. SELECT m.title, b.rating * 10 AS rating_percent FROM movies m JOIN boxoffice b ON m.id = b.movie_id;
3. SELECT title, year FROM movies WHERE year % 2 = 0;

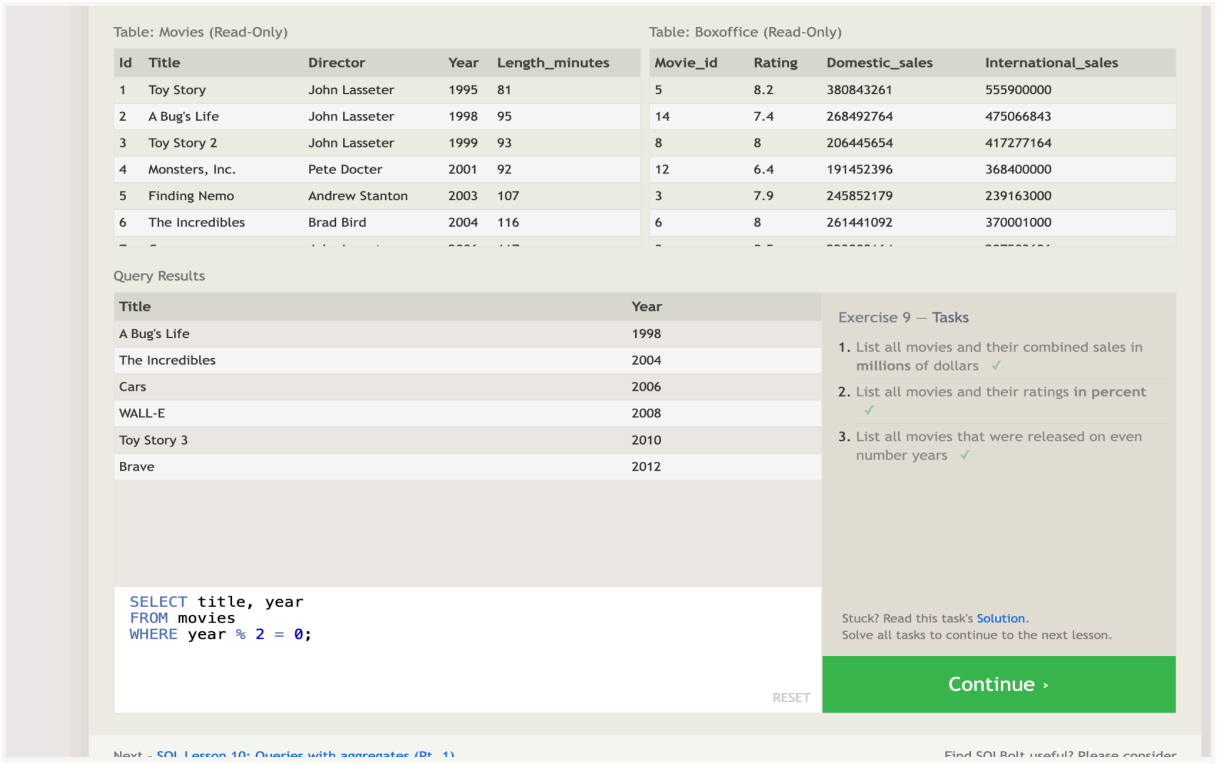


Table: Movies (Read-Only)

Id	Title	Director	Year	Length_minutes
1	Toy Story	John Lasseter	1995	81
2	A Bug's Life	John Lasseter	1998	95
3	Toy Story 2	John Lasseter	1999	93
4	Monsters, Inc.	Pete Docter	2001	92
5	Finding Nemo	Andrew Stanton	2003	107
6	The Incredibles	Brad Bird	2004	116

Table: Boxoffice (Read-Only)

Movie_id	Rating	Domestic_sales	International_sales
5	8.2	380843261	555900000
14	7.4	268492764	475066843
8	8	206445654	417277164
12	6.4	191452396	368400000
3	7.9	245852179	239163000
6	8	261441092	370001000

Query Results

Title	Year
A Bug's Life	1998
The Incredibles	2004
Cars	2006
WALL-E	2008
Toy Story 3	2010
Brave	2012

Exercise 9 — Tasks

1. List all movies and their combined sales in **millions** of dollars ✓
2. List all movies and their ratings **in percent** ✓
3. List all movies that were released on even number years ✓

```
SELECT title, year
FROM movies
WHERE year % 2 = 0;
```

RESET

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

Continue ›

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

Find SQL Bolt useful? Please consider

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

1. SELECT MAX(years_employed) AS longest_time_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_worked FROM employees GROUP BY building;

```
FROM mytable
WHERE constraint_expression
GROUP BY column;
```

The **GROUP BY** clause works by grouping rows that have the same value in the column specified.

Exercise

For this exercise, we are going to work with our **Employees** table. Notice how the rows in this table have shared data, which will give us an opportunity to use aggregate functions to summarize some high-level metrics about the teams. Go ahead and give it a shot.

Table: Employees

Building	Total_years_worked
1e	29
2w	36

```
SELECT building,
       SUM(years_employed) AS total_years_worked
FROM employees
GROUP BY building;
;
```

RESET

Exercise 10 – Tasks

1. Find the longest time that an employee has been at the studio ✓
2. For each role, find the average number of years employed by employees in that role ✓
3. Find the total number of employee years worked in each building ✓

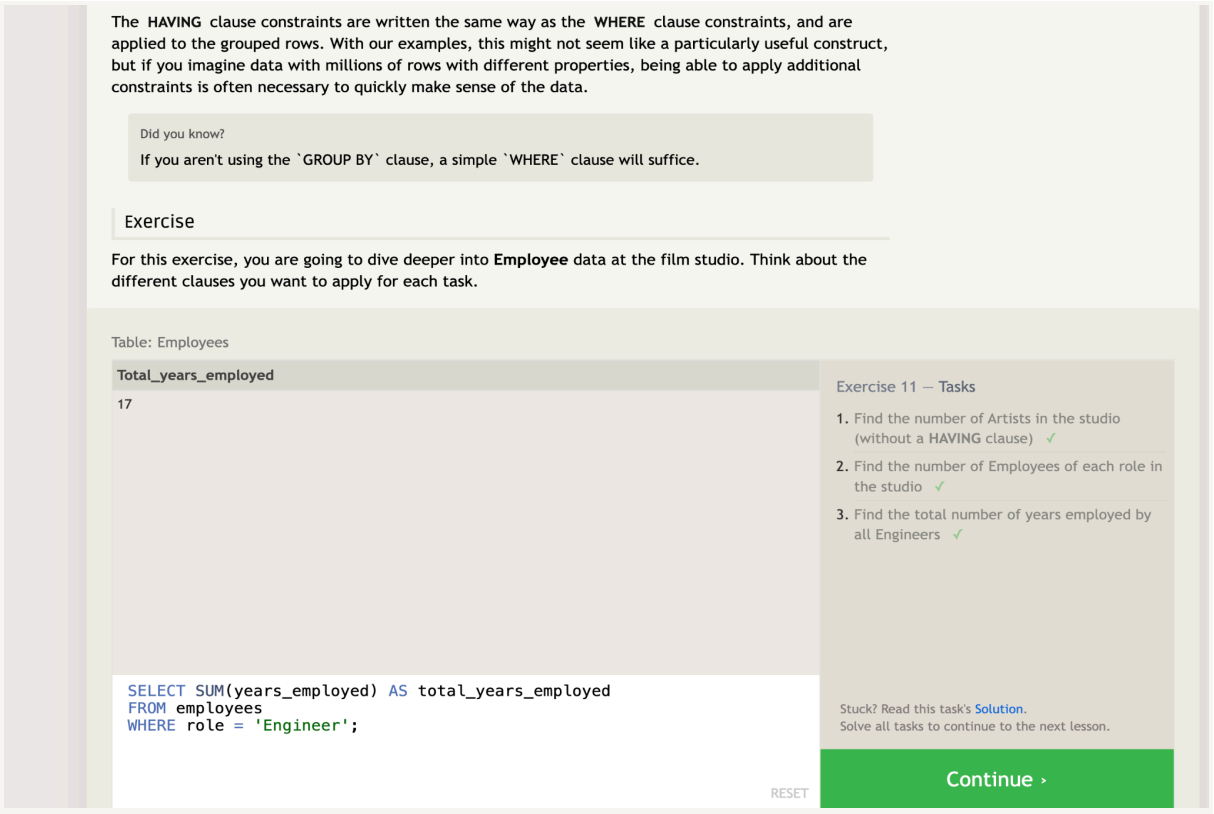
Stuck? Read this task's Solution.
Solve all tasks to continue to the next lesson.

Continue ›

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

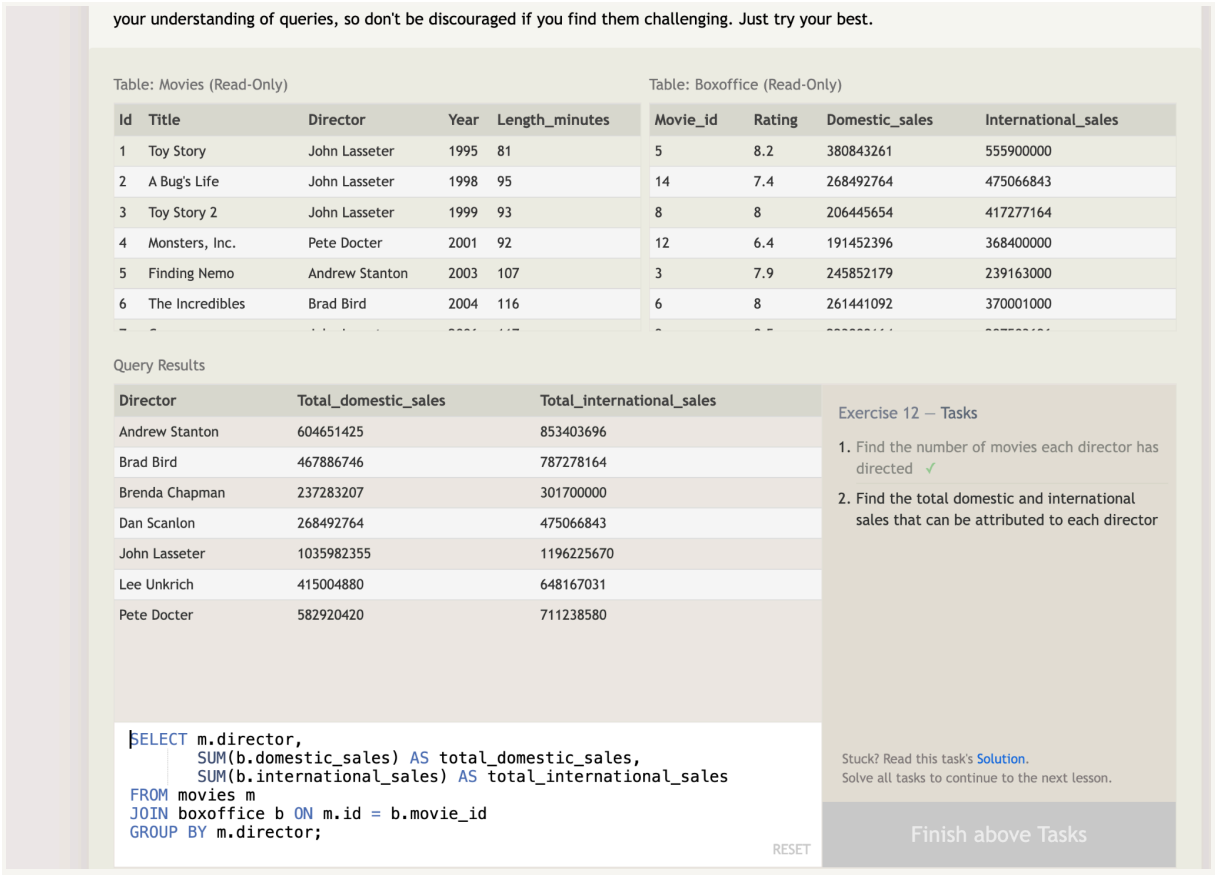
SQL Lesson 11: Queries with aggregates (Pt. 2)

1. SELECT COUNT(*) AS number_of_artists FROM employees WHERE role = 'Artist';
2. SELECT role, COUNT(*) AS number_of_employees FROM employees GROUP BY role;
3. SELECT SUM(years_employed) AS total_years_employed FROM employees WHERE role = 'Engineer';



SQL Lesson 12: Order of execution of a Query

1. SELECT m.director, COUNT(*) AS number_of_movies FROM movies m GROUP BY m.director;



your understanding of queries, so don't be discouraged if you find them challenging. Just try your best.

Table: Movies (Read-Only)

Id	Title	Director	Year	Length_minutes
1	Toy Story	John Lasseter	1995	81
2	A Bug's Life	John Lasseter	1998	95
3	Toy Story 2	John Lasseter	1999	93
4	Monsters, Inc.	Pete Docter	2001	92
5	Finding Nemo	Andrew Stanton	2003	107
6	The Incredibles	Brad Bird	2004	116

Table: Boxoffice (Read-Only)

Movie_id	Rating	Domestic_sales	International_sales
5	8.2	380843261	555900000
14	7.4	268492764	475066843
8	8	206445654	417277164
12	6.4	191452396	368400000
3	7.9	245852179	239163000
6	8	261441092	370001000

Query Results

Director	Total_domestic_sales	Total_international_sales
Andrew Stanton	604651425	853403696
Brad Bird	467886746	787278164
Brenda Chapman	237283207	301700000
Dan Scanlon	268492764	475066843
John Lasseter	1035982355	1196225670
Lee Unkrich	415004880	648167031
Pete Docter	582920420	711238580

```
SELECT m.director,
       SUM(b.domestic_sales) AS total_domestic_sales,
       SUM(b.international_sales) AS total_international_sales
FROM movies m
JOIN boxoffice b ON m.id = b.movie_id
GROUP BY m.director;
```

RESET

Exercise 12 – Tasks

1. Find the number of movies each director has directed ✓
2. Find the total domestic and international sales that can be attributed to each director

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

Finish above Tasks

SQL Lesson 13: Inserting rows

1. INSERT INTO movies (title, director, year, length_minutes) VALUES ('Toy Story 4', 'Josh Cooley', 2019, 100);
2. INSERT INTO boxoffice (movie_id, rating, domestic_sales, international_sales) VALUES (15, 8.7, 340000000, 270000000);

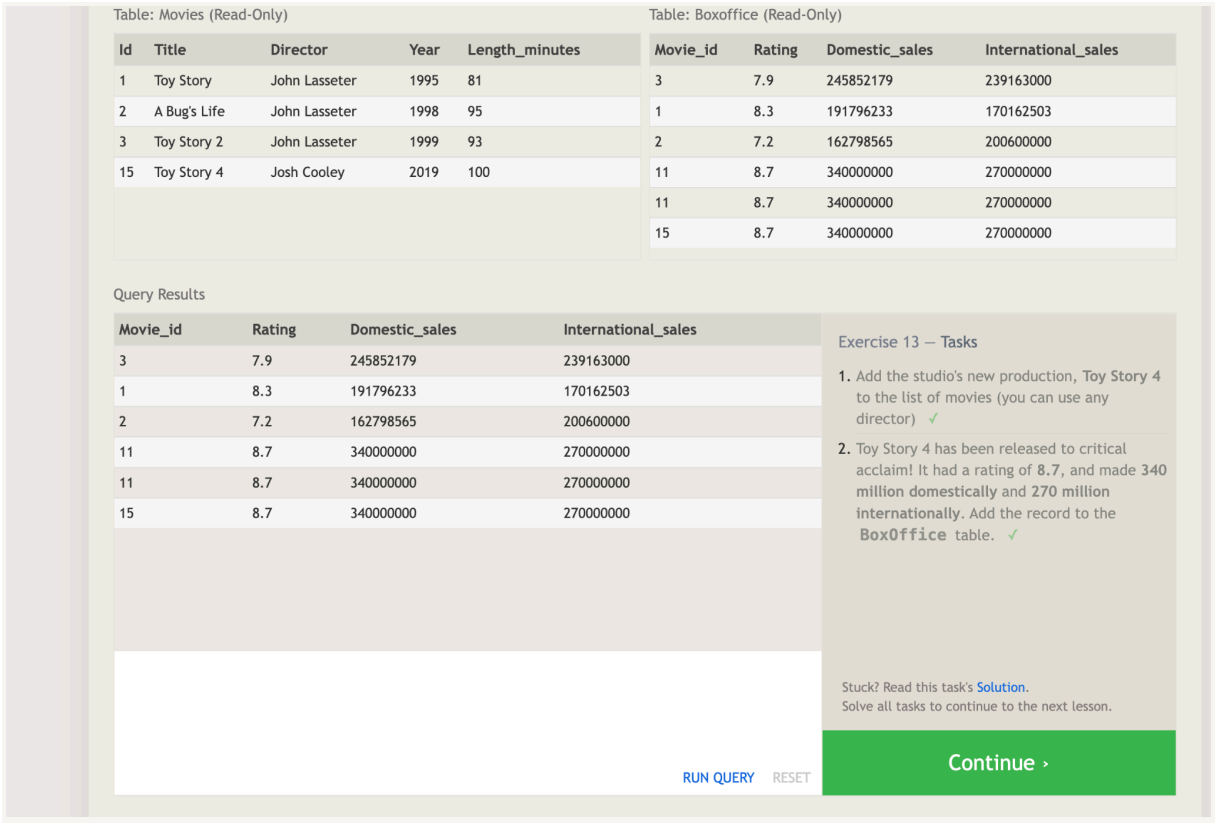


Table: Movies (Read-Only)

Id	Title	Director	Year	Length_minutes
1	Toy Story	John Lasseter	1995	81
2	A Bug's Life	John Lasseter	1998	95
3	Toy Story 2	John Lasseter	1999	93
15	Toy Story 4	Josh Cooley	2019	100

Table: Boxoffice (Read-Only)

Movie_id	Rating	Domestic_sales	International_sales
3	7.9	245852179	239163000
1	8.3	191796233	170162503
2	7.2	162798565	200600000
11	8.7	340000000	270000000
11	8.7	340000000	270000000
15	8.7	340000000	270000000

Query Results

Movie_id	Rating	Domestic_sales	International_sales
3	7.9	245852179	239163000
1	8.3	191796233	170162503
2	7.2	162798565	200600000
11	8.7	340000000	270000000
11	8.7	340000000	270000000
15	8.7	340000000	270000000

Exercise 13 — Tasks

1. Add the studio's new production, **Toy Story 4** to the list of movies (you can use any director) ✓
2. Toy Story 4 has been released to critical acclaim! It had a rating of **8.7**, and made **340 million domestically** and **270 million internationally**. Add the record to the `BoxOffice` table. ✓

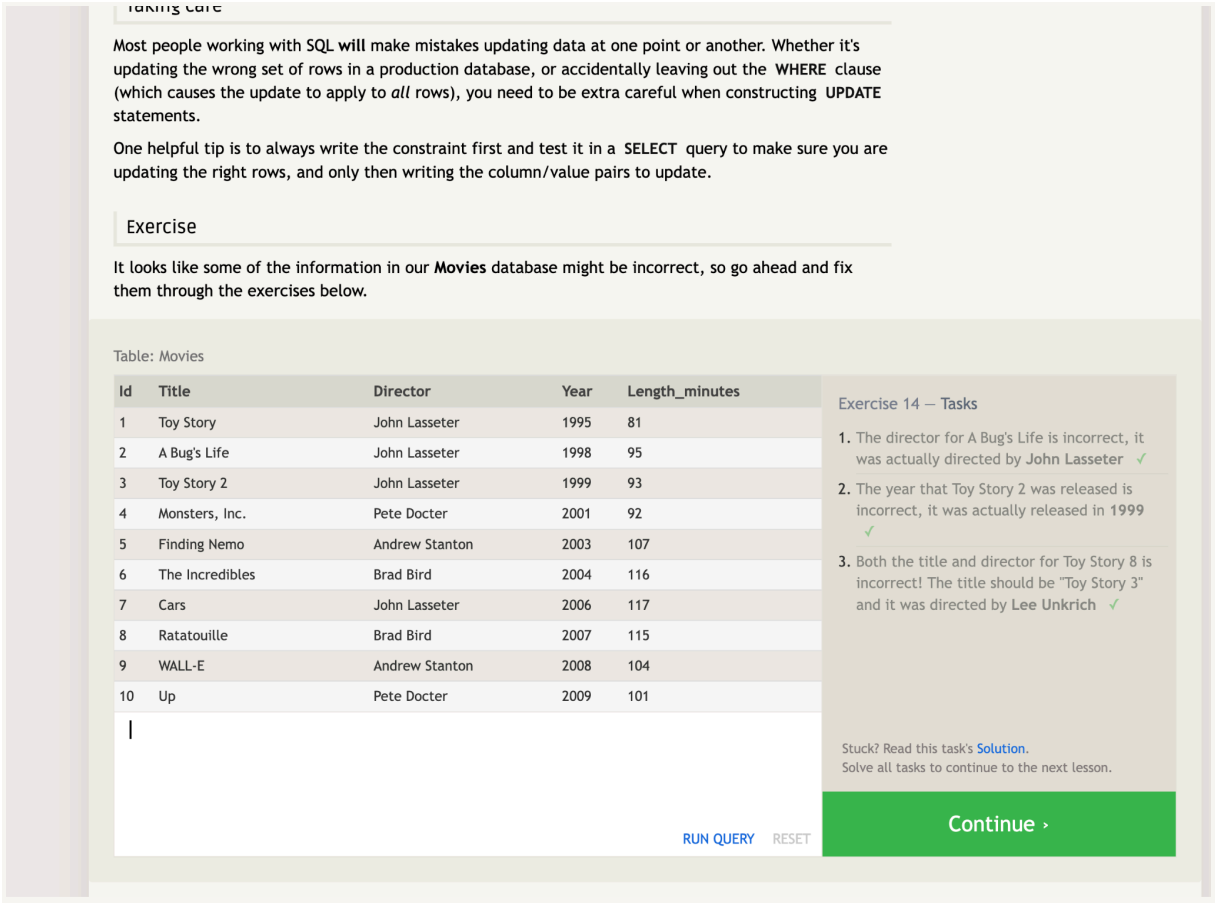
Stuck? Read this task's Solution.
Solve all tasks to continue to the next lesson.

RUN QUERY RESET

Continue ›

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;



Most people working with SQL **will** make mistakes updating data at one point or another. Whether it's updating the wrong set of rows in a production database, or accidentally leaving out the **WHERE** clause (which causes the update to apply to *all* rows), you need to be extra careful when constructing **UPDATE** statements.

One helpful tip is to always write the constraint first and test it in a **SELECT** query to make sure you are updating the right rows, and only then writing the column/value pairs to update.

Exercise

It looks like some of the information in our **Movies** database might be incorrect, so go ahead and fix them through the exercises below.

Table: Movies

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

Exercise 14 — Tasks

1. The director for A Bug's Life is incorrect, it was actually directed by **John Lasseter** ✓
2. The year that Toy Story 2 was released is incorrect, it was actually released in **1999** ✓
3. Both the title and director for Toy Story 8 is incorrect! The title should be "Toy Story 3" and it was directed by **Lee Unkrich** ✓

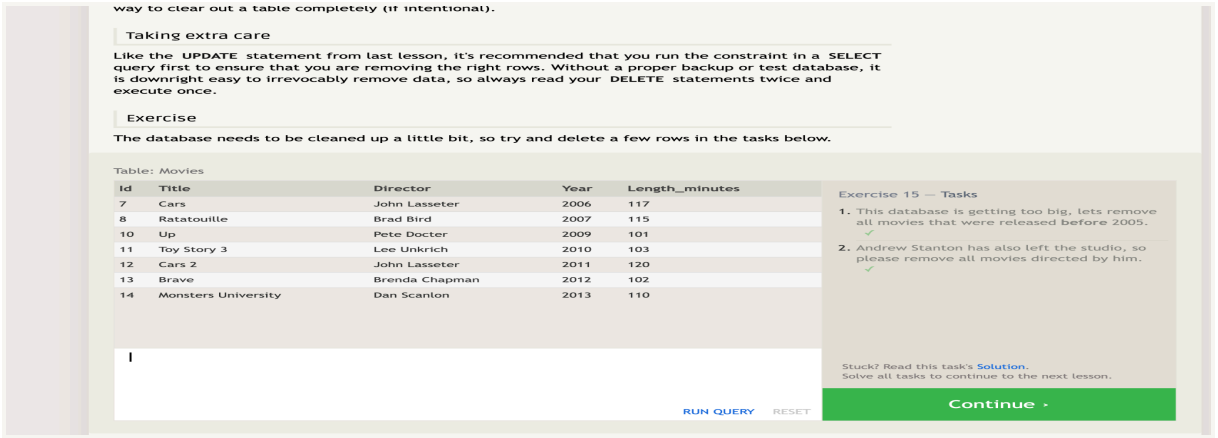
Stuck? Read this task's Solution.
Solve all tasks to continue to the next lesson.

RUN QUERY RESET

Continue ›

SQL Lesson 15: Deleting rows

1. DELETE FROM movies WHERE year < 2005;
2. DELETE FROM m



way to clear out a table completely (if intentional).

Taking extra care

Like the **UPDATE** statement from last lesson, it's recommended that you run the constraint in a **SELECT** query first to ensure that you are removing the right rows. Without a proper backup or test database, it is downright easy to irrevocably remove data, so always read your **DELETE** statements twice and execute once.

Exercise

The database needs to be cleaned up a little bit, so try and delete a few rows in the tasks below.

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

Exercise 15 — Tasks

1. This database is getting too big, lets remove all movies that were released before 2005. ✓
2. Andrew Stanton has also left the studio, so please remove all movies directed by him. ✓

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

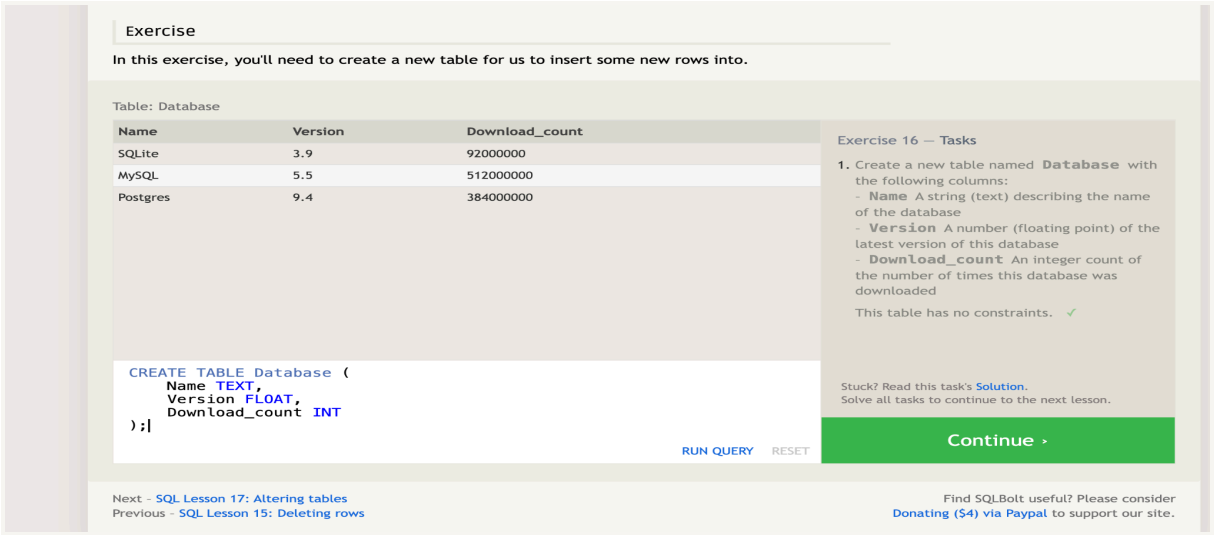
RUN QUERY RESET

Continue ›

ovies WHERE director = 'Andrew Stanton';

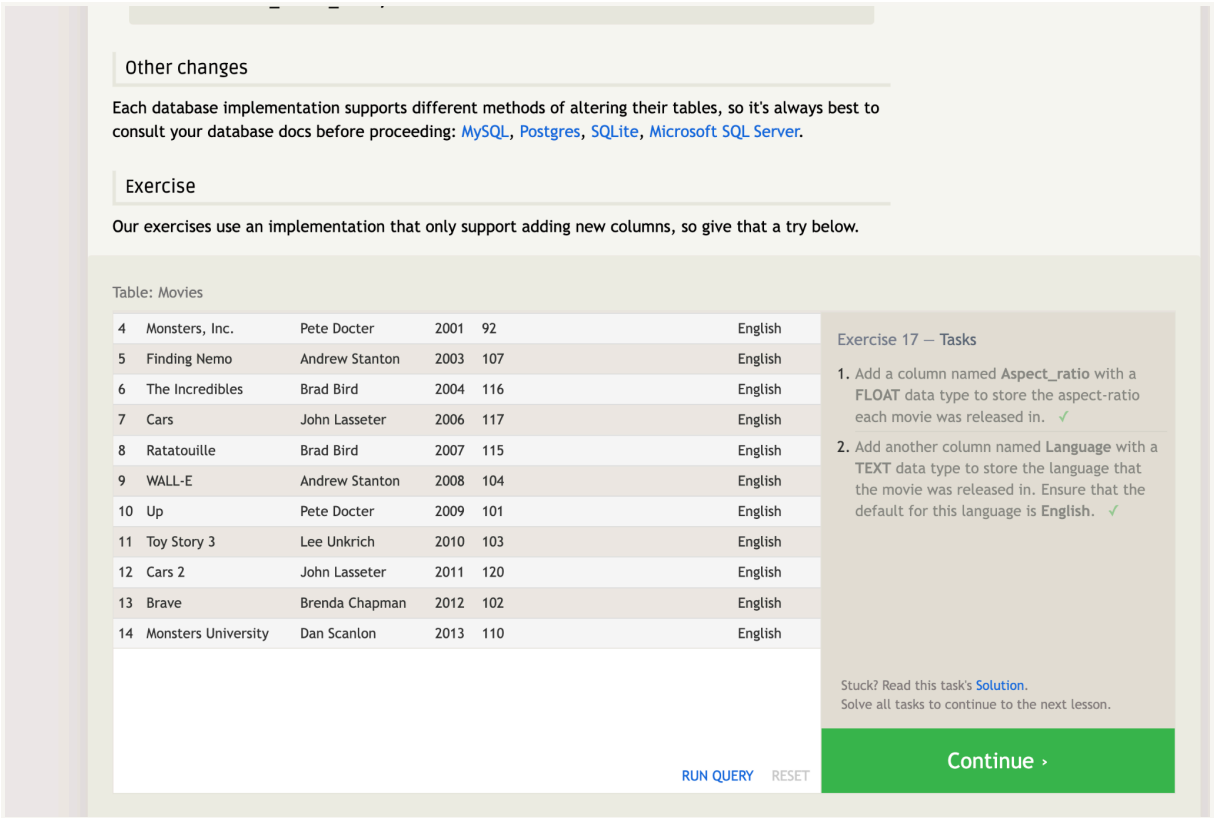
SQL Lesson 16: Creating tables

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



SQL Lesson 17: Altering tables

1. ALTER TABLE movies ADD COLUMN Aspect_ratio FLOAT;
2. ALTER TABLE movies ADD COLUMN Language TEXT DEFAULT 'English';



SQL Lesson 18: Dropping tables

1. DROP TABLE movies;

2. DROP TABLE boxoffice;