SQL Task - Day 1

SQL Lesson 1: SELECT queries 101

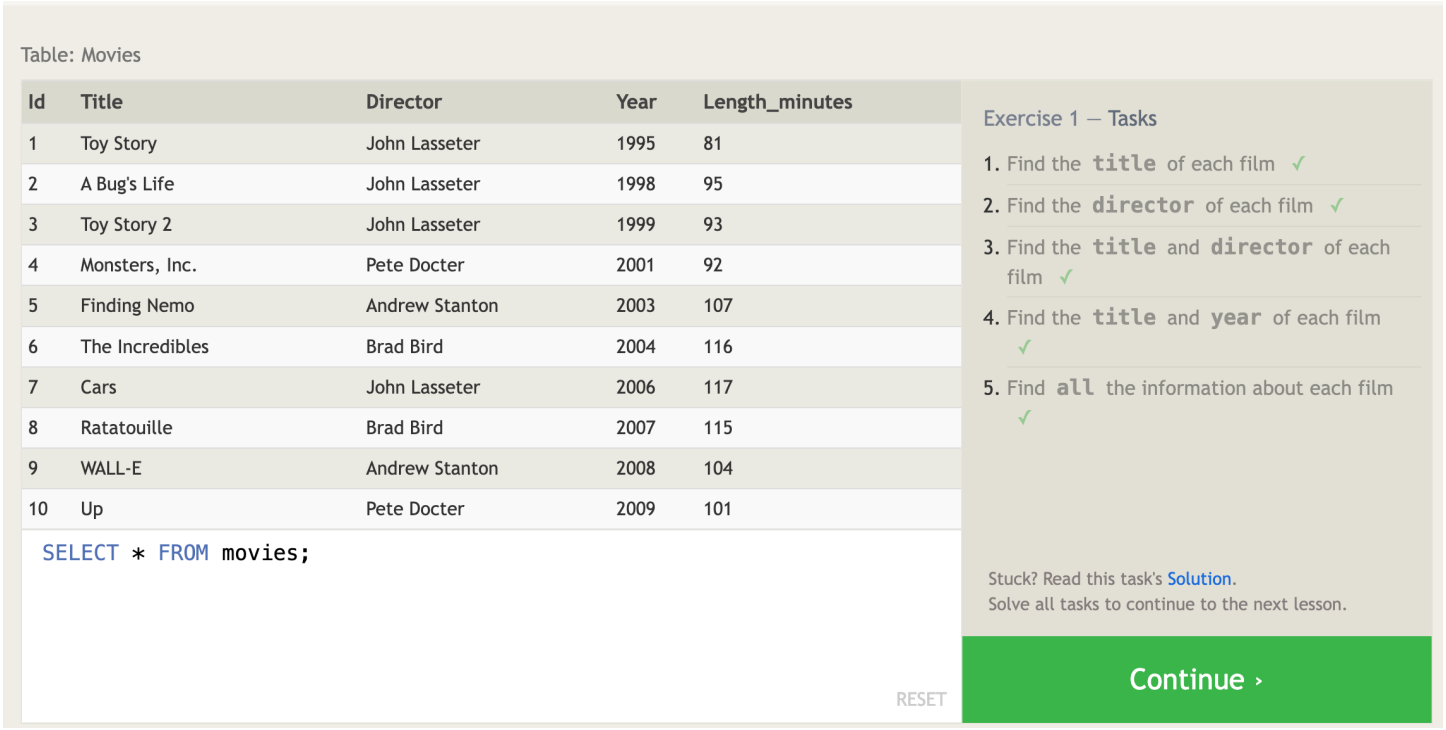


Table: Movies

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

```
SELECT * FROM movies;
```

RESET

Exercise 1 – Tasks

1. Find the `title` of each film ✓
2. Find the `director` of each film ✓
3. Find the `title` and `director` of each film ✓
4. Find the `title` and `year` of each film ✓
5. Find `all` the information about each film ✓

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

Continue ›

Queries:

- SELECT title FROM movies;
- SELECT director FROM movies;
- SELECT title and director FROM movies;
- SELECT title and year FROM movies;
- SELECT * FROM movies;

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

Exercise

Using the right constraints, find the information we need from the **Movies** table for each task below.

Table: Movies

Title	Year
Toy Story	1995
A Bug's Life	1998
Toy Story 2	1999
Monsters, Inc.	2001
Finding Nemo	2003

```
SELECT title, year FROM movies
WHERE year <= 2003;
```

RESET

Exercise 2 — Tasks

1. Find the movie with a row `id` of 6 ✓
2. Find the movies released in the `year`s between 2000 and 2010 ✓
3. Find the movies **not** released in the `year`s between 2000 and 2010 ✓
4. Find the first 5 Pixar movies and their release `year` ✓

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

Continue ›

Queries:

- SELECT title FROM movies where id=6;
- SELECT * FROM movies where year between 2000 and 2010;
- SELECT * FROM movies where year not between 2000 and 2010;
- SELECT title,year FROM movies where id between 1 and 5;

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

Table: Movies

Title	Director
WALL-E	Andrew Stanton
WALL-G	Brenda Chapman

```
SELECT title, director FROM movies
WHERE TITLE LIKE "WALL%"
```

RESET

Exercise 3 — Tasks

1. Find all the Toy Story movies ✓
2. Find all the movies directed by John Lasseter ✓
3. Find all the movies (and director) not directed by John Lasseter ✓
4. Find all the WALL-* movies ✓

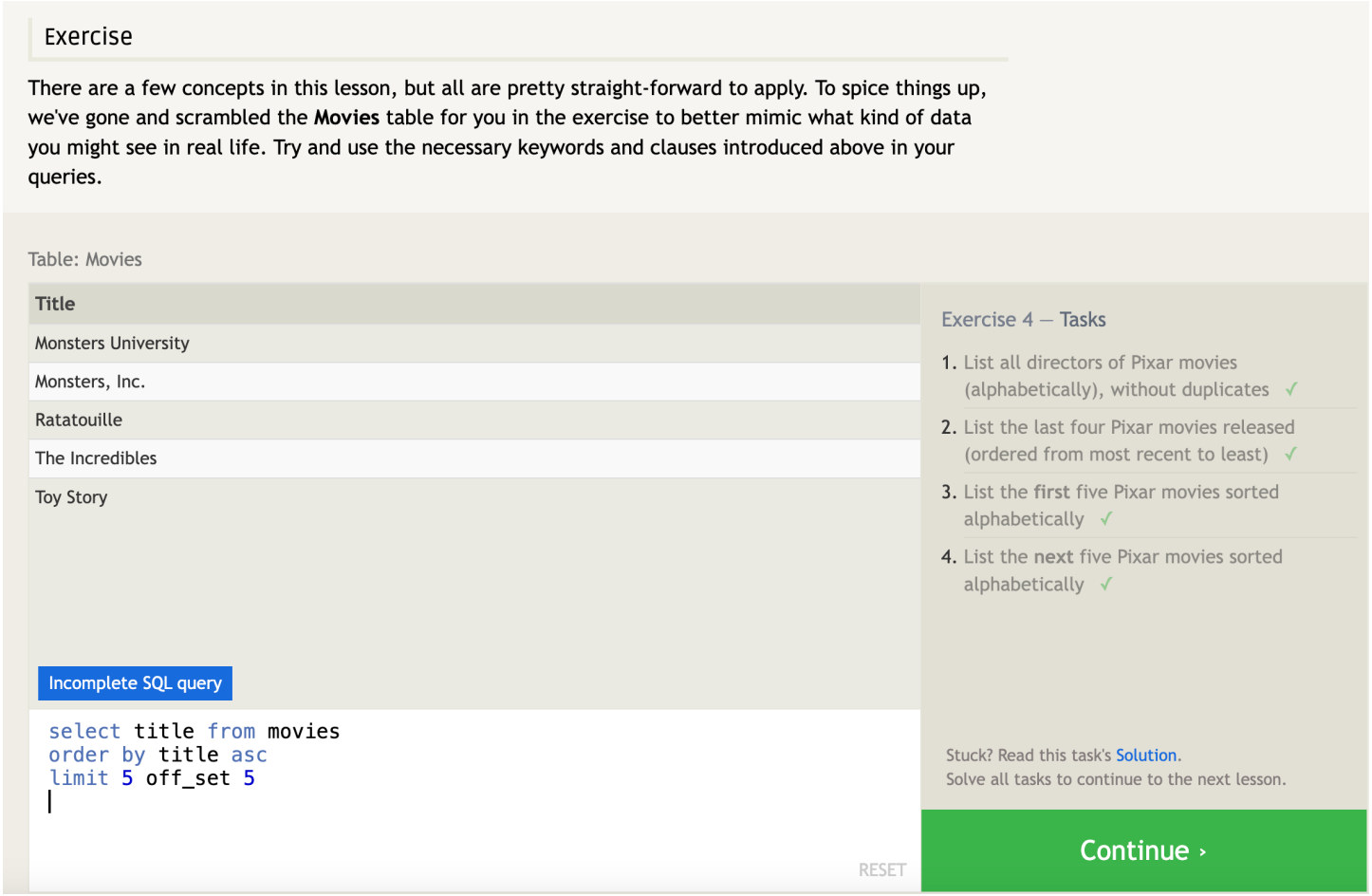
Stuck? Read this task's Solution.
Solve all tasks to continue to the next lesson.

Continue ›

Queries:

- SELECT * FROM movies where title in ("Toy Story","Toy Story 2","Toy Story 3");
- SELECT * FROM movies where director = 'John Lasseter';
- SELECT * FROM movies where director != 'John Lasseter';
- SELECT title, director FROM movies where TITLE LIKE 'WALL%';

SQL Lesson 4: Filtering and sorting Query results



Queries:

- SELECT * FROM movies group by director order by director asc;
- SELECT * FROM movies order by year desc limit 4;
- SELECT * FROM movies order by title asc limit 5;
- SELECT title FROM movies order by title asc limit 5 offset 5;

SQL Review: Simple SELECT Queries

Try and write some queries to find the information requested in the tasks you know. You may have to use a different combination of clauses in your query for each task. Once you're done, continue onto the next lesson to learn about queries that span multiple tables.

Table: North_american_cities

City	Population
Chicago	2718782
Houston	2195914

```
SELECT city, population FROM north_american_cities
WHERE country LIKE "United States"
ORDER BY population DESC
LIMIT 2 OFFSET 2;
```

RESET

Review 1 — Tasks

1. List all the Canadian cities and their populations ✓
2. Order all the cities in the United States by their latitude from north to south ✓
3. List all the cities west of Chicago, ordered from west to east ✓
4. List the two largest cities in Mexico (by population) ✓
5. List the third and fourth largest cities (by population) in the United States and their population ✓

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

Continue ›

Queries:

- SELECT * FROM north_american_cities where country = 'Canada';
- SELECT * FROM north_american_cities where country= 'United States'order by latitude desc;
- SELECT * FROM north_american_cities where longitude < (select longitude from north_american_cities where city = 'Chicago') order by longitude asc;
- SELECT * FROM north_american_cities where country = 'Mexico' order by population desc limit 2;
- 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

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 title, rating
FROM movies
  JOIN boxoffice
    ON movies.id = boxoffice.movie_id
ORDER BY 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 ›

Queries:

- SELECT * FROM movies join boxoffice on movies.id = boxoffice.movie_id;
- SELECT * FROM movies join boxoffice on movies.id = boxoffice.movie_id where international_sales > domestic_sales;
- SELECT title,rating FROM movies JOIN boxoffice ON movies.id = boxoffice.movie_id ORDER BY rating DESC;

SQL Lesson 7: OUTER JOINs

Query Results

Building_name	Role
1e	Engineer
1e	Manager
1w	
2e	
2w	Artist
2w	Manager

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

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) ✓

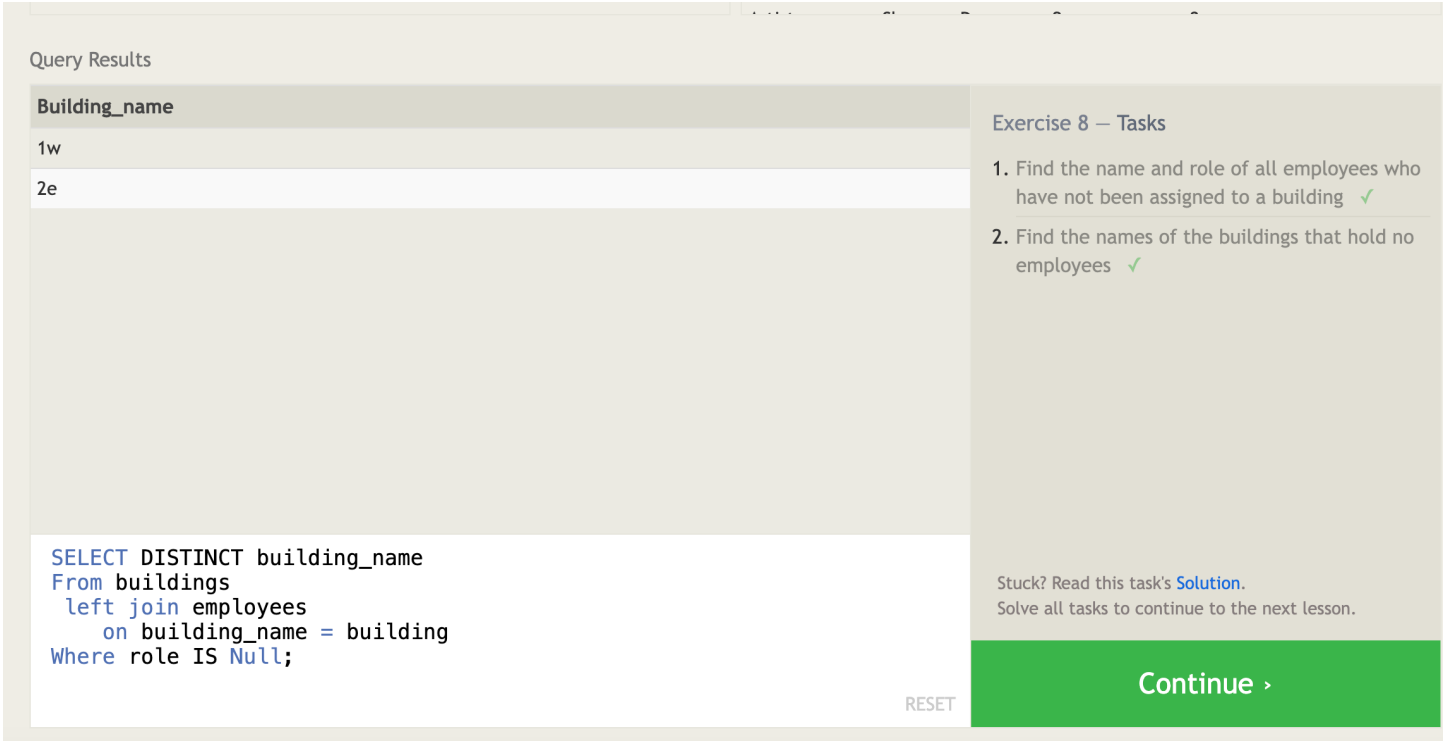
Stuck? Read this task's Solution.
Solve all tasks to continue to the next lesson.

Continue ›

Queries:

- SELECT building FROM employees join buildings on buildings.building_name = employees.building group by building;
- SELECT building_name,capacity FROM buildings;
- SELECT DISTINCT building_name, role FROM buildings LEFT JOIN employees ON building_name = building;

SQL Lesson 8: A short note on NULLs



Queries:

- SELECT name,role FROM employees where building is null;
- SELECT distinct building_name From buildings left join employees on building_name=building where role is not null;

SQL Lesson 9: Queries with expressions

Query Results

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

```
SELECT title, year
from movies
where year % 2 = 0;
```

RESET

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 ✓

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

Continue ›

Queries:

- SELECT title, (domestic_sales + international_sales) / 1000000 AS millions FROM movies JOIN boxoffice ON movies.id = boxoffice.movie_id;
- SELECT title, (rating) * 10 AS Ratings FROM movies JOIN boxoffice ON movies.id = boxoffice.movie_id;
- SELECT title, year from movies where year%2==0;

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

Table: Employees

Building	Total_years_employed
1e	29
2w	36

```
SELECT building, sum(years_employed) as Total_years_employed
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.

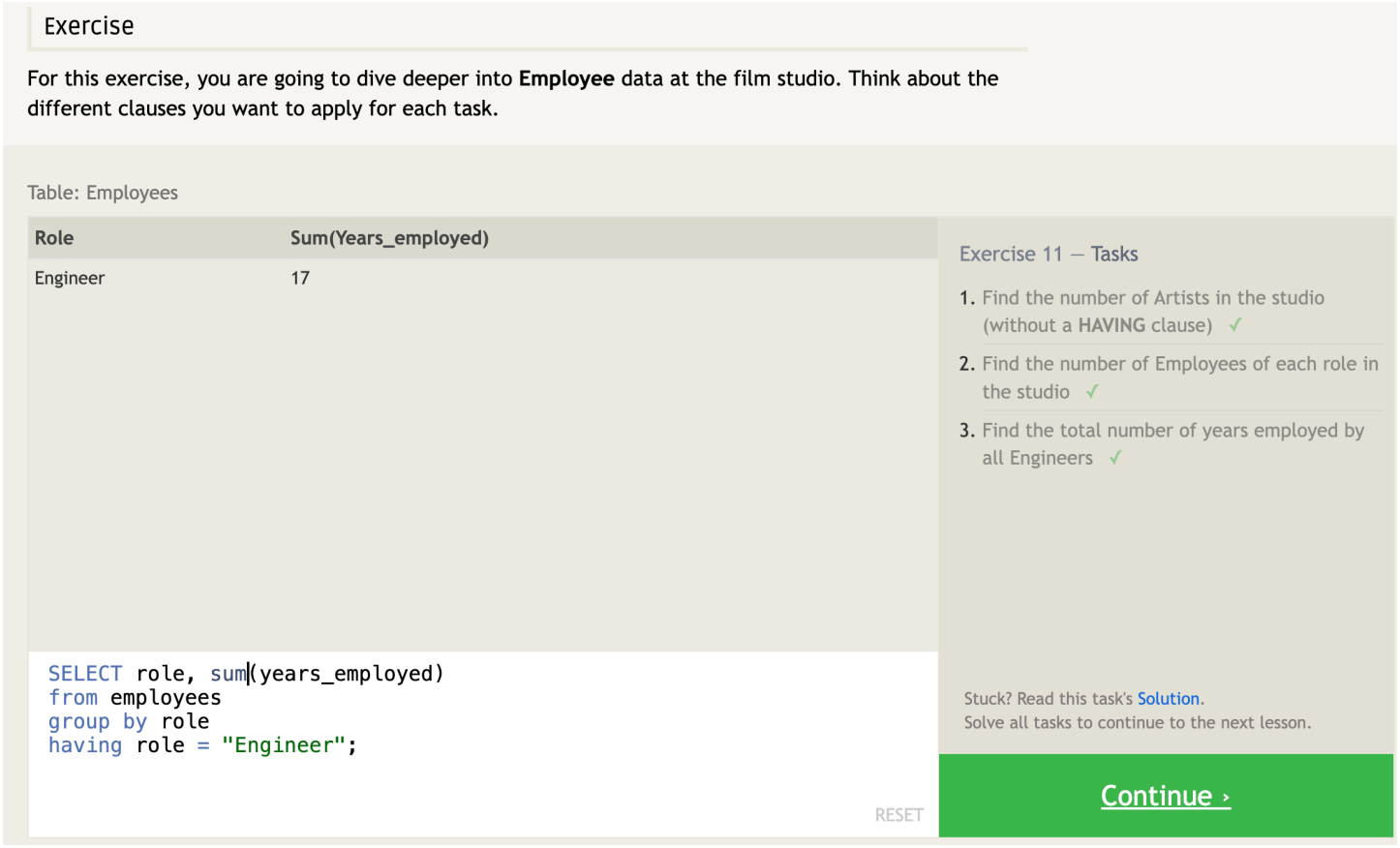
Continue ›

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

Queries:

- SELECT max(years_employed) FROM employees;
- SELECT role,avg(years_employed) as average FROM employees group by role;
- SELECT building,sum(years_employed) as Total_years_employed from employees group by building;

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



Queries:

- SELECT count(role) FROM employees where role = 'Artist' ;
- select role,count(role) from employees group by role;
- SELECT role, sum(years_employed) from employees group by role having role = "Engineer";

SQL Lesson 12: Order of execution of a Query

Query Results

Director	Cumulative_sales_from_all_movies
Andrew Stanton	1458055121
Brad Bird	1255164910
Brenda Chapman	538983207
Dan Scanlon	743559607
John Lasseter	2232208025
Lee Unkrich	1063171911
Pete Docter	1294159000

```
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;
```

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.

Continue ›

Next - SQL Lesson 13: Inserting rows

Find SQLBolt useful? Please consider

Queries:

- SELECT Director, count(director) FROM movies group by director;
- SELECT director, sum (domestic_sales + International_sales) as total from movies join boxoffice on id=movie_id group by Director ;

SQL Lesson 13: Inserting rows

Query Results

Movie_id	Rating	Domestic_sales	International_sales
3	7.9	245852179	239163000
1	8.3	191796233	170162503
2	7.2	162798565	200600000
4	8.7	340000000	270000000

```
insert into boxoffice values (4, 8.7, 340000000, 270000000);
```

RUN QUERY RESET

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.

Continue ›

Next - SQL Lesson 14: Updating rows
Previous - SQL Lesson 12: Order of execution of a Query

Find SQLBolt useful? Please consider
Donating ($4) via Paypal to support our site

Queries:

- insert into movies VALUES (4, "Toy Story 4", "Pete Docter", 2015, 90);
- insert into boxoffice values (4, 8.7, 340000000, 270000000);

SQL Lesson 14: Updating rows

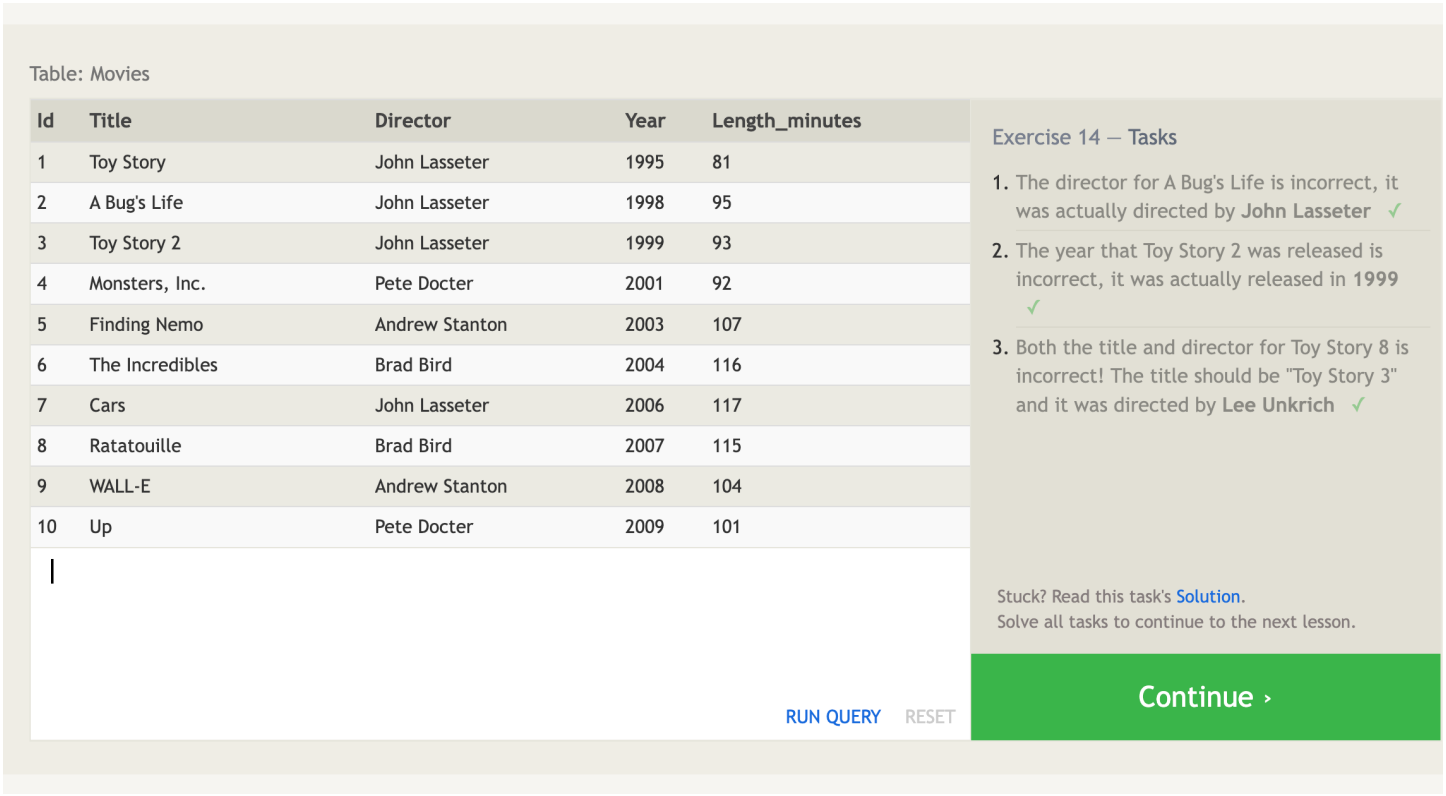


Table: Movies

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

RUN QUERY RESET

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.

Continue ›

Queries:

- update Movies set Director = 'John Lasseter' where Title = "A Bug's Life"
- update Movies set year = 1999 where Title = "Toy Story 2"
- update Movies set Title = "Toy Story 3",Director='Lee Unkrich' where Title = "Toy Story 8"

SQL Lesson 15: Deleting rows

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

```
delete from movies
where director = "Andrew Stanton";
```

RUN QUERY RESET

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. ✓

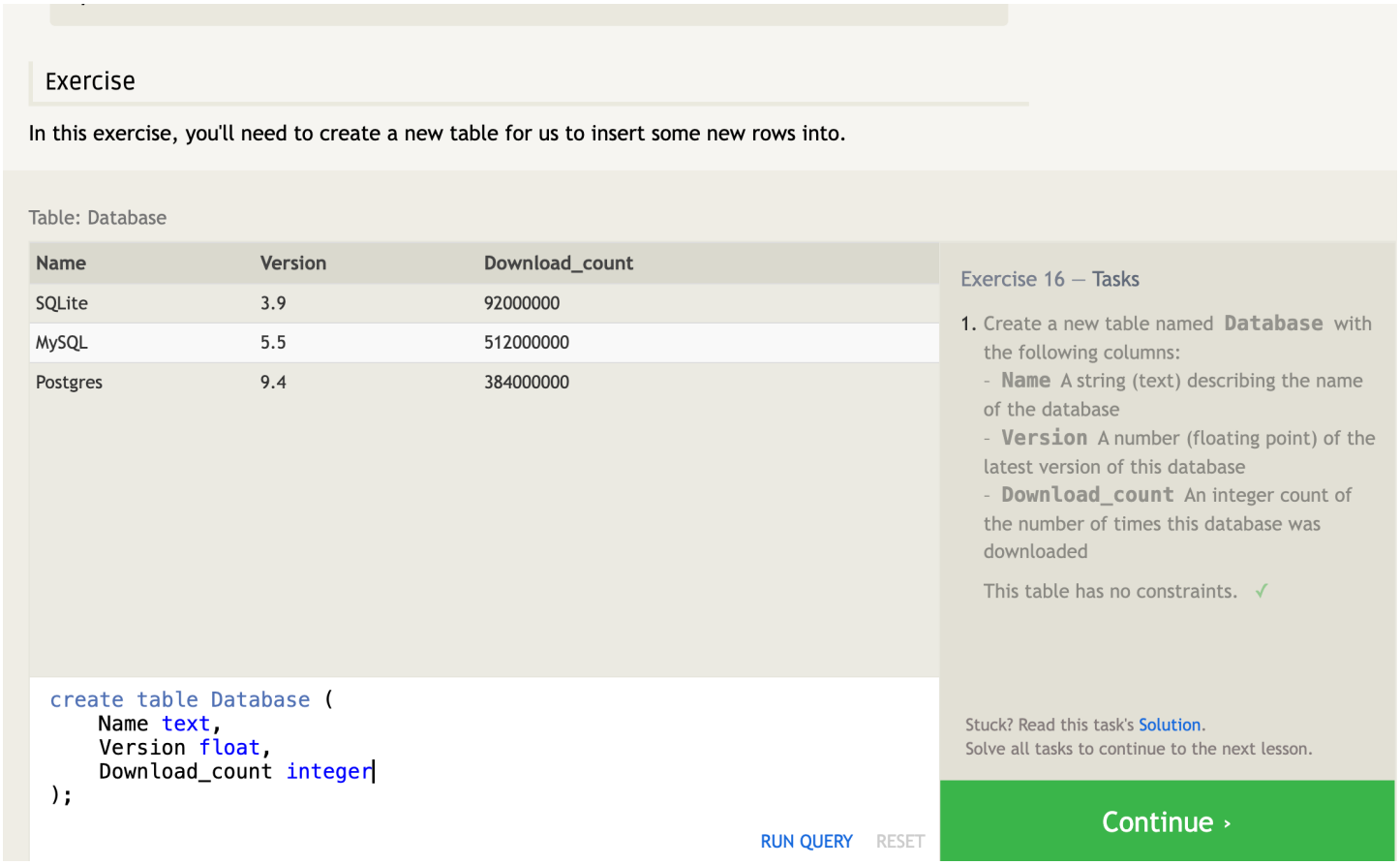
Stuck? Read this task's Solution.
Solve all tasks to continue to the next lesson.

Continue ›

Queries:

- delete from Movies where year<2005;
- delete from movies where director=”Andrew Santon”;

SQL Lesson 16: Creating tables



Queries:

- create table Database (

Name text,

Version float,

Download_count int

)

SQL Lesson 17: Altering tables

Exercise

Our exercises use an implementation that only support adding new columns, so give that a try below.

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

```
alter table Movies
 add column Language text default "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**. ✓

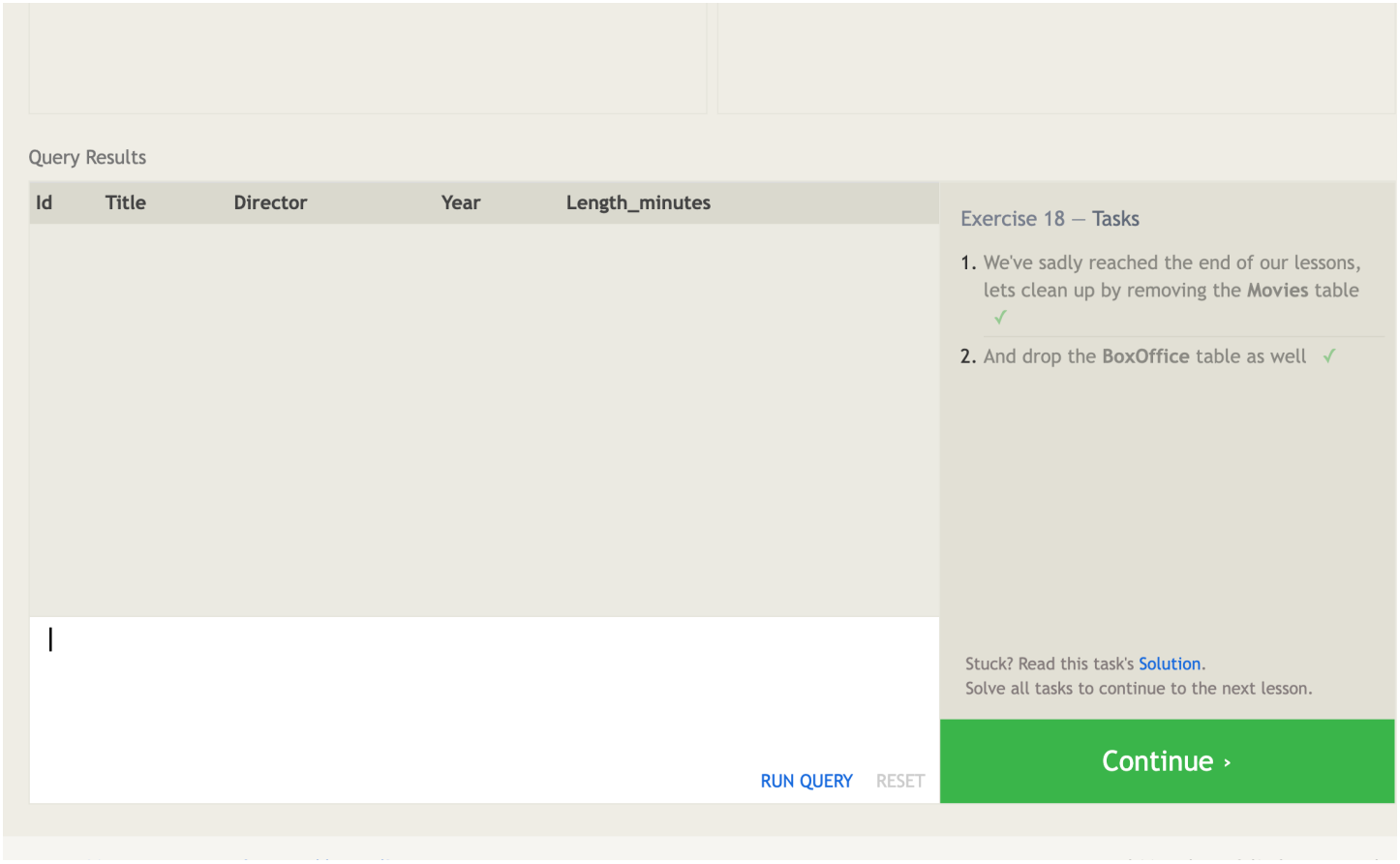
Stuck? Read this task's Solution.
Solve all tasks to continue to the next lesson.

Continue ›

Queries:

- alter table Movies add Aspect_ratio float;
- alter table Movies add Language text default English;

SQL Lesson 18: Dropping tables



Queries:

- drop table if exists Movies;
- drop table if exists BoxOffice;