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;

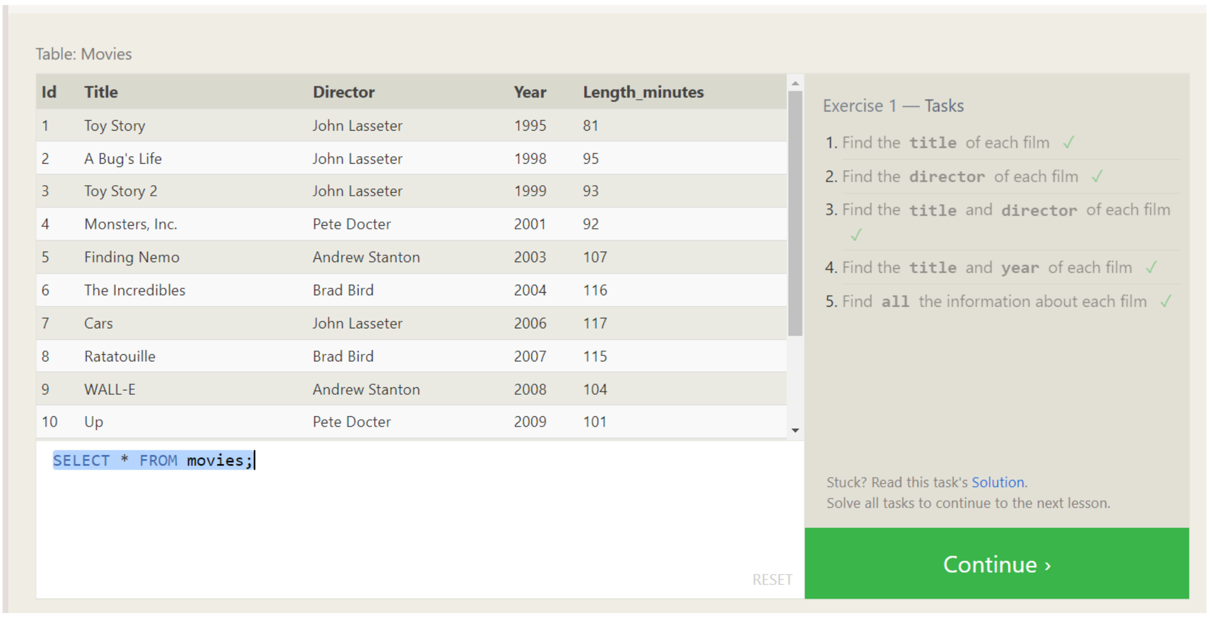


Table: Movies

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

```
SELECT * FROM movies;
```

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.

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;

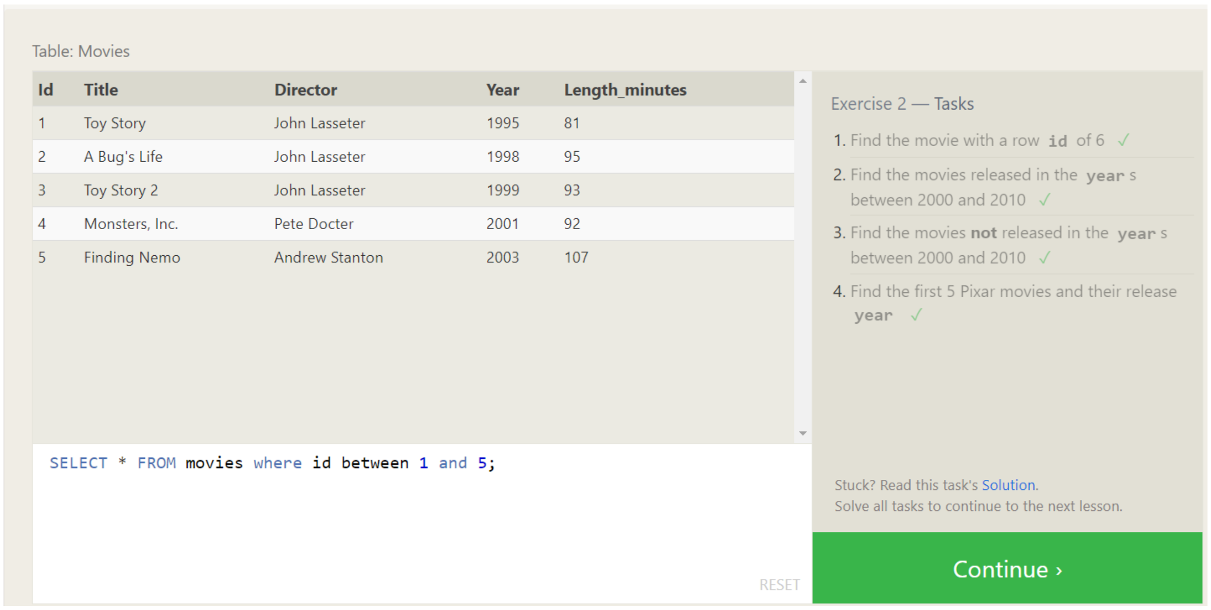


Table: Movies

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

```
SELECT * FROM movies where id between 1 and 5;
```

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.

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%";

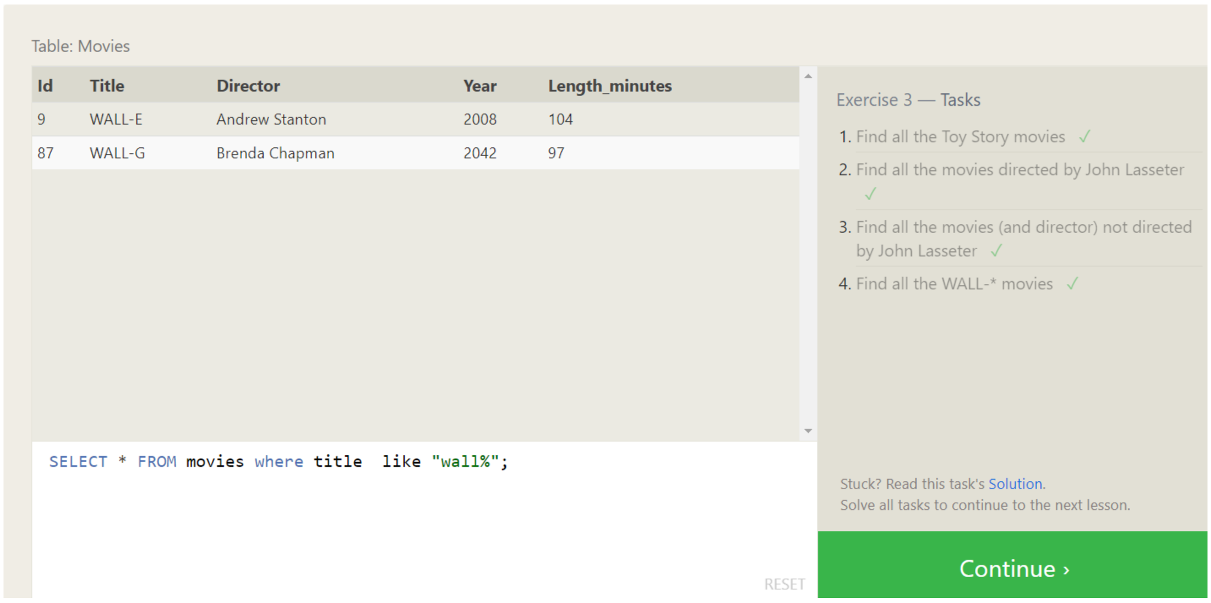


Table: Movies

Id	Title	Director	Year	Length_minutes
9	WALL-E	Andrew Stanton	2008	104
87	WALL-G	Brenda Chapman	2042	97

```
SELECT * FROM movies where title  like "wall%";
```

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.

SQL Lesson 4: Filtering and sorting Query results

1. SELECT distinct director FROM movies order by director;
2. SELECT * FROM movies order by year desc limit 4;
3. SELECT * FROM movies order by title limit 5;
4. SELECT * FROM movies order by title limit 5 offset 5;

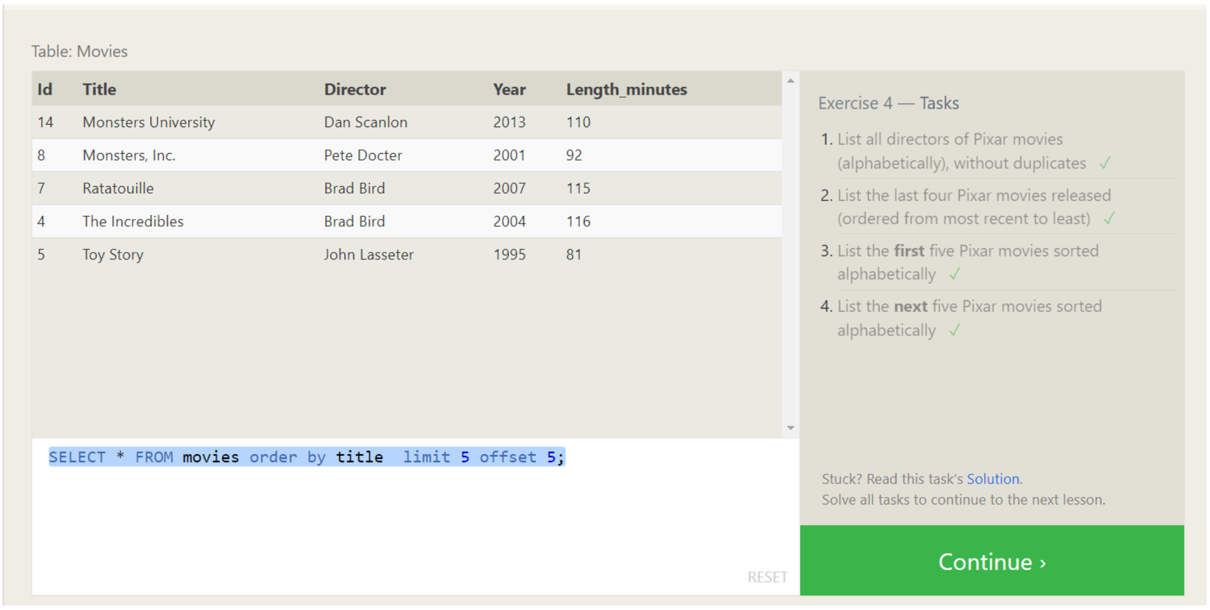


Table: Movies

Id	Title	Director	Year	Length_minutes
14	Monsters University	Dan Scanlon	2013	110
8	Monsters, Inc.	Pete Docter	2001	92
7	Ratatouille	Brad Bird	2007	115
4	The Incredibles	Brad Bird	2004	116
5	Toy Story	John Lasseter	1995	81

```
SELECT * FROM movies order by title  limit 5 offset 5;
```

Exercise 4 — Tasks

1. List all directors of Pixar movies (alphabetically), without duplicates ✓
2. List the last four Pixar movies released (ordered from most recent to least) ✓
3. List the **first** five Pixar movies sorted alphabetically ✓
4. List the **next** five Pixar movies sorted alphabetically ✓

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

SQL Review: Simple SELECT Queries

1. SELECT city, population FROM north_american_cities where country like "canada";
2. select city, latitude from north_american_cities where country like "united states" order by latitude desc;
3. SELECT * FROM north_american_cities where longitude < -87.629798 order by longitude asc;
4. SELECT * FROM north_american_cities where country like "mexico" order by population desc limit 2
5. SELECT * FROM north_american_cities where country like "united states" order by population desc limit 2 offset 2;
6.

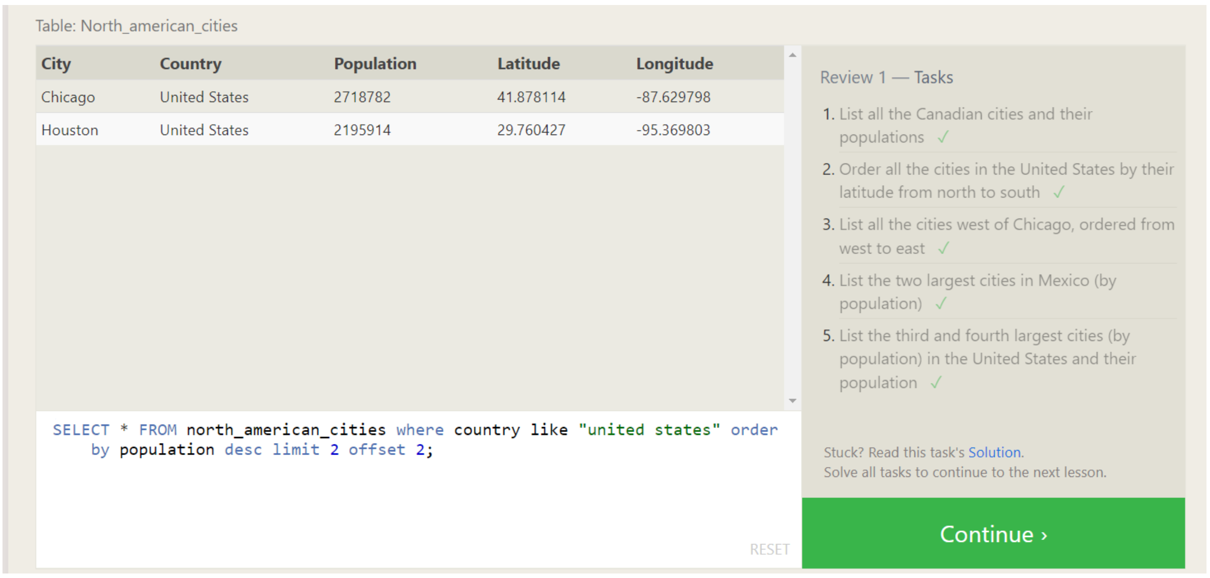


Table: North_american_cities

City	Country	Population	Latitude	Longitude
Chicago	United States	2718782	41.878114	-87.629798
Houston	United States	2195914	29.760427	-95.369803

```
SELECT * FROM north_american_cities where country like "united states" order
    by population desc limit 2 offset 2;
```

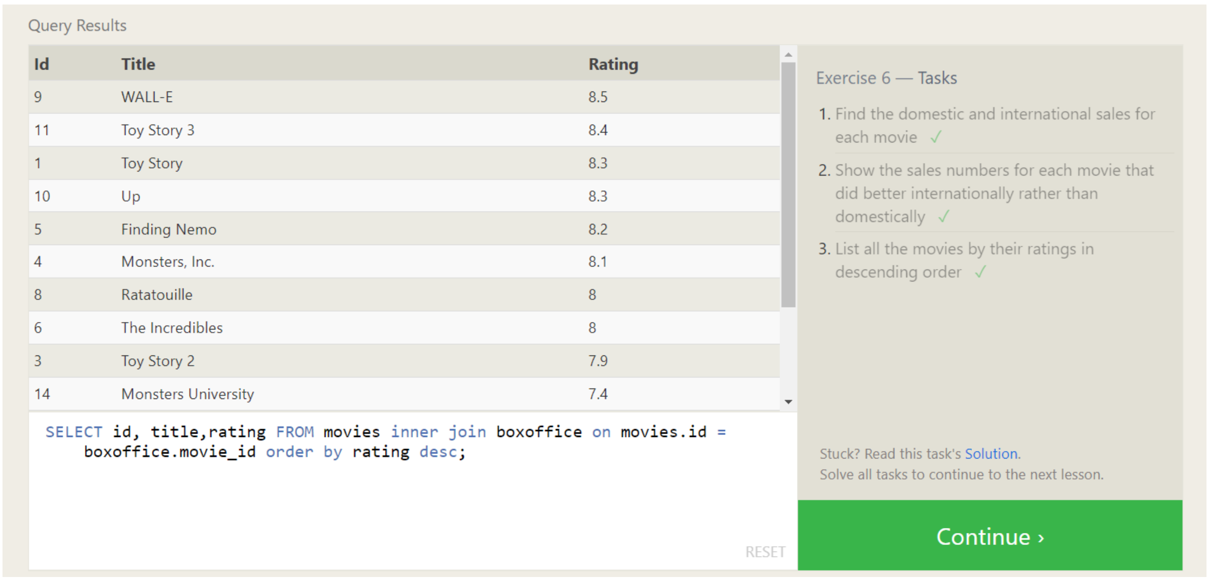
Review 1 — Tasks

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.

SQL Lesson 6: Multi-table queries with JOINs

1. SELECT id, title, domestic_sales, international_sales FROM movies inner join boxoffice on movies.id = boxoffice.movie_id;
2. SELECT id, title, domestic_sales, international_sales FROM movies inner join boxoffice on movies.id = boxoffice.movie_id where international_sales > domestic_sales;
3. SELECT id, title,rating FROM movies inner join boxoffice on movies.id = boxoffice.movie_id order by rating desc;


Query Results

Id	Title	Rating
9	WALL-E	8.5
11	Toy Story 3	8.4
1	Toy Story	8.3
10	Up	8.3
5	Finding Nemo	8.2
4	Monsters, Inc.	8.1
8	Ratatouille	8
6	The Incredibles	8
3	Toy Story 2	7.9
14	Monsters University	7.4

```
SELECT id, title,rating FROM movies inner join boxoffice on movies.id =
    boxoffice.movie_id order by rating desc;
```

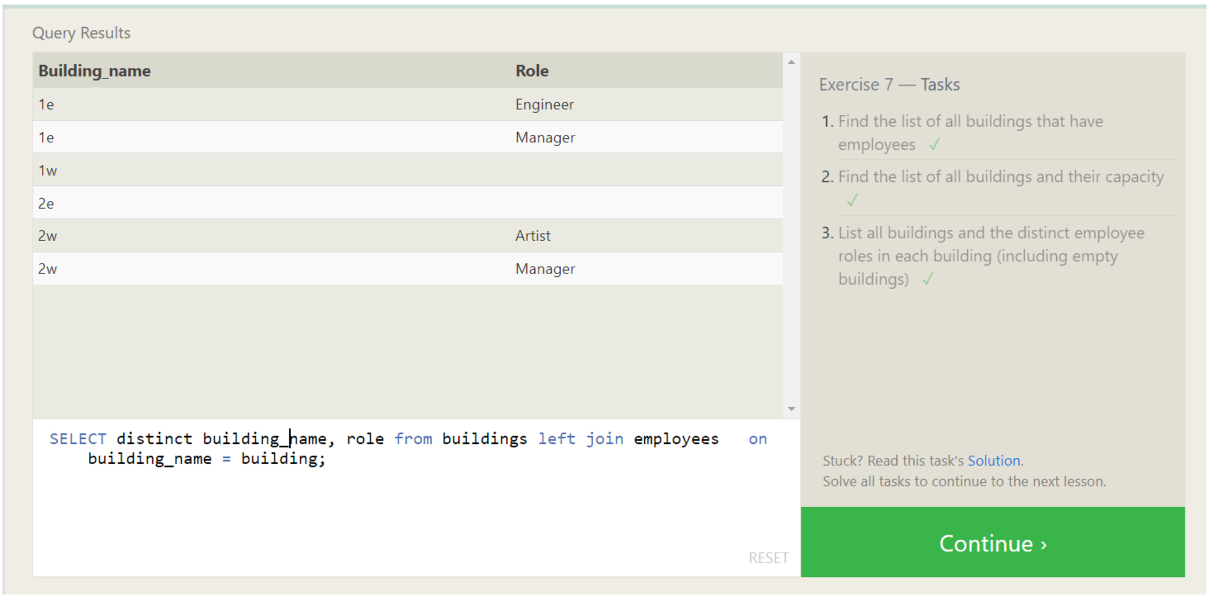
Exercise 6 — Tasks

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.

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;


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;
```

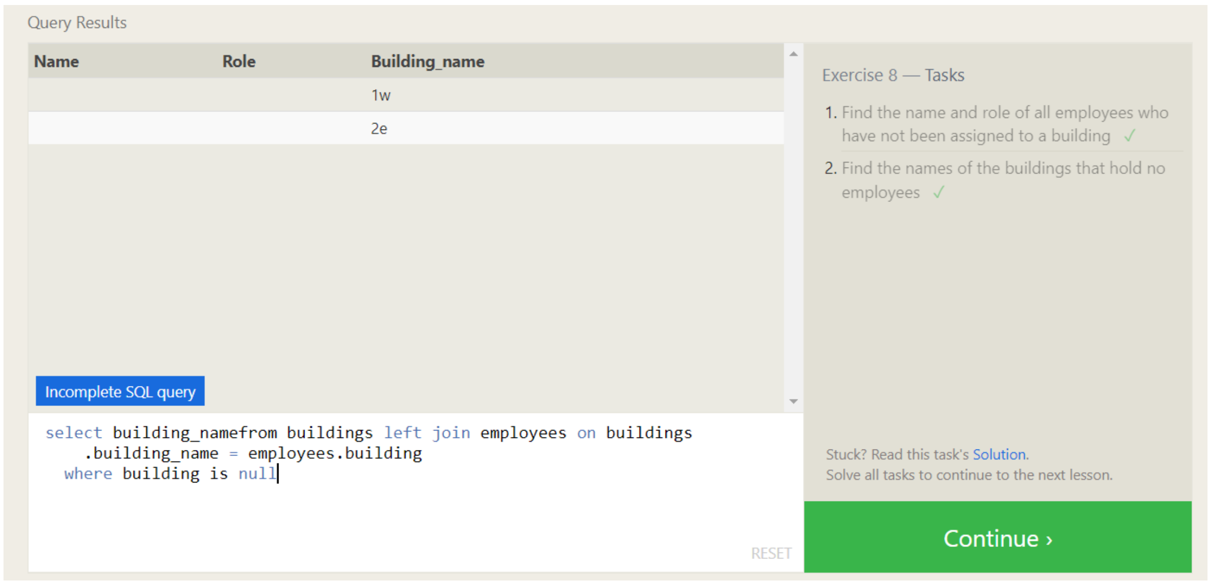
Exercise 7 — Tasks

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.

SQL Lesson 8: A short note on NULLs

1. SELECT name, role FROM employees where building is null
2. select building_namefrom buildings left join employees on buildings.building_name employees.building where building is null;


Query Results

Name	Role	Building_name
		1w
		2e

```
select building_namefrom buildings left join employees on buildings
    .building_name = employees.building
  where building is null
```

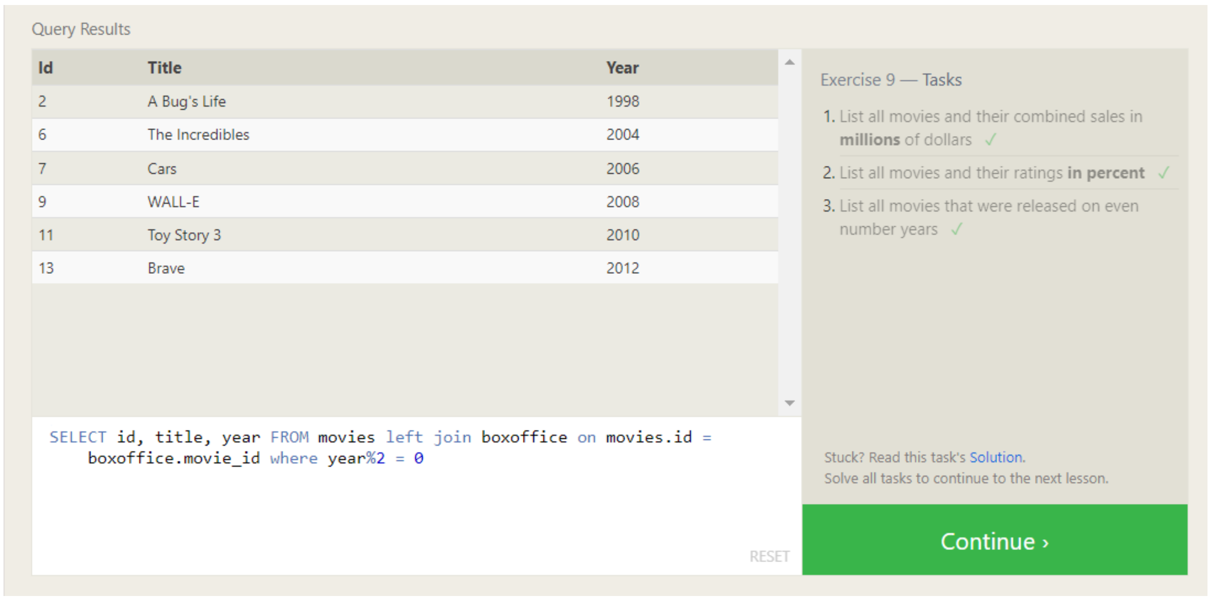
Exercise 8 — Tasks

1. Find the name and role of all employees who have not been assigned to a building ✓
2. Find the names of the buildings that hold no employees ✓

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

SQL Lesson 9: Queries with expressions

1. SELECT id, title, (domestic_sales + international_sales)/1000000 as Total_sales FROM movies left join boxoffice on movies.id = boxoffice.movie_id
2. SELECT id, title, (rating*10) as Ratings_percent FROM movies left join boxoffice on movies.id = boxoffice.movie_id
3. SELECT id, title, year FROM movies left join boxoffice on movies.id = boxoffice.movie_id where year%2 = 0


Query Results

Id	Title	Year
2	A Bug's Life	1998
6	The Incredibles	2004
7	Cars	2006
9	WALL-E	2008
11	Toy Story 3	2010
13	Brave	2012

```
SELECT id, title, year FROM movies left join boxoffice on movies.id =
    boxoffice.movie_id where year%2 = 0
```

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.

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

1. SELECT name, max(years_employed) FROM employees;
2. SELECT role, avg(years_employed) FROM employees group by role.
3. SELECT building, sum(years_employed) FROM employees group by building

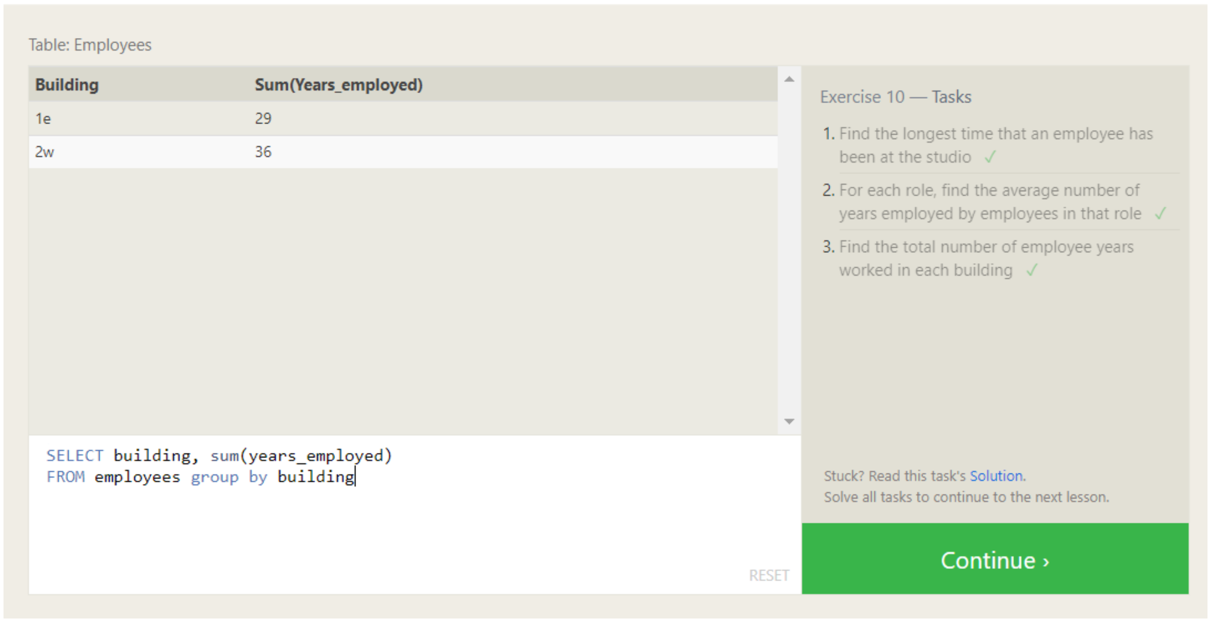


Table: Employees

Building	Sum(Years_employed)
1e	29
2w	36

```
SELECT building, sum(years_employed)
FROM employees group by building
```

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.

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

1. SELECT count(role) FROM employees where role like "artist"
2. SELECT role, count(name) from employees group by role
3. select role, sum(years_employed) from employees where role like "engineer"

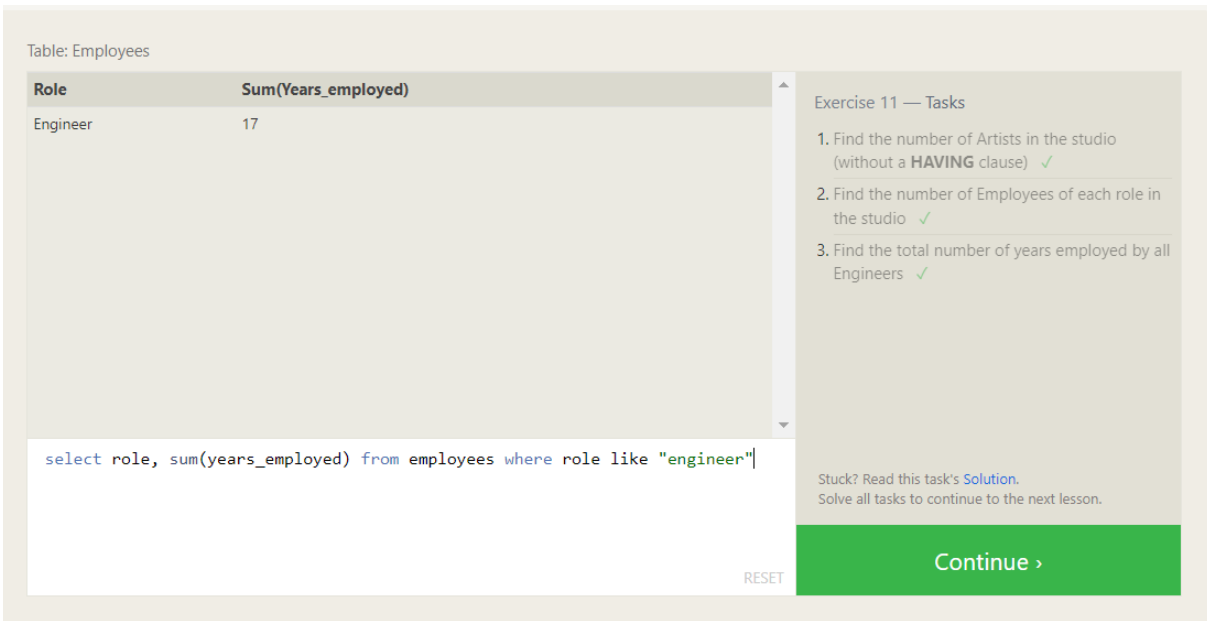


Table: Employees

Role	Sum(Years_employed)
Engineer	17

```
select role, sum(years_employed) from employees where role like "engineer"
```

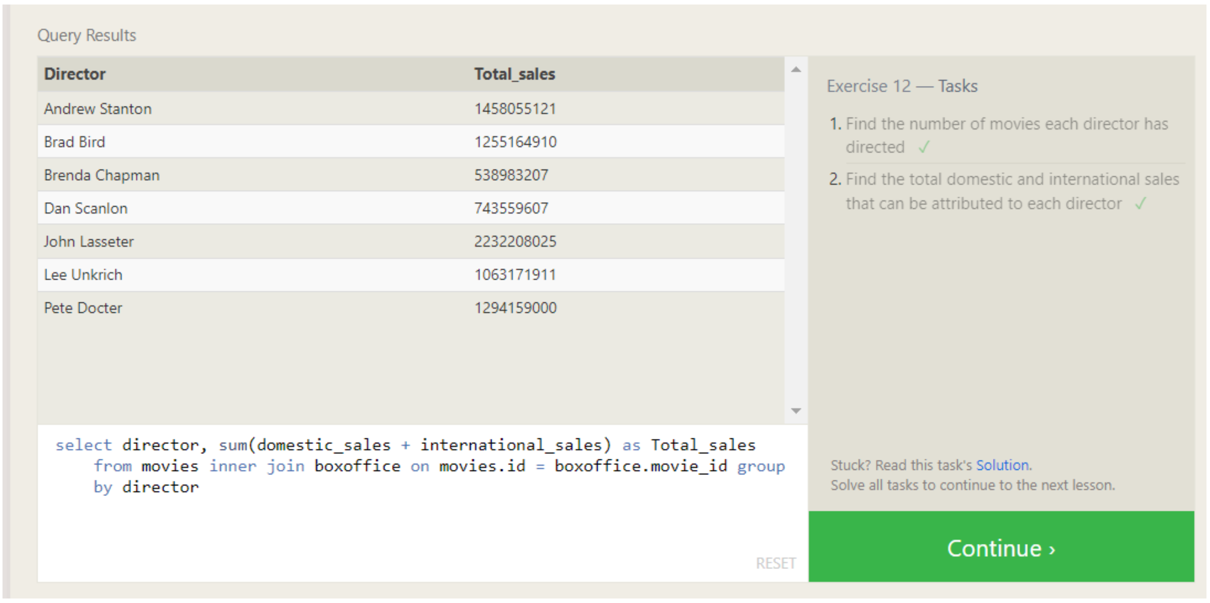
Exercise 11 — Tasks

1. Find the number of Artists in the studio (without a **HAVING** clause) ✓
2. Find the number of Employees of each role in the studio ✓
3. Find the total number of years employed by all Engineers ✓

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

SQL Lesson 12: Order of execution of a Query

1. SELECT director, count(director) as number_of_Movies FROM movies group by director
2. select director, sum(domestic_sales + international_sales) as Total_sales from movies inner join boxoffice on movies.id = boxoffice.movie_id group by director



Query Results

Director	Total_sales
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 Total_sales
    from movies inner join boxoffice on movies.id = boxoffice.movie_id group
    by director
```

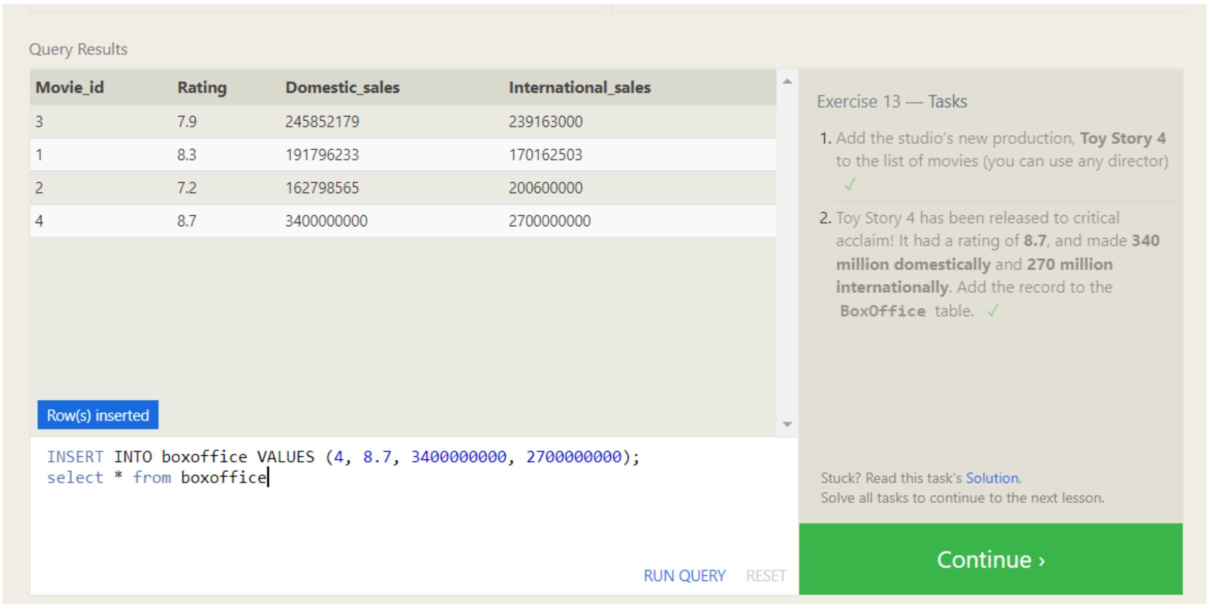
Exercise 12 — Tasks

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.

SQL Lesson 13: Inserting rows

1. INSERT INTO movies VALUES (4, "Toy Story 4", " John Lasseter", 2012, 112);
2. INSERT INTO boxoffice VALUES (4, 8.7, 3400000000, 2700000000);


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	3400000000	2700000000

Row(s) inserted

```
INSERT INTO boxoffice VALUES (4, 8.7, 3400000000, 2700000000);
select * from boxoffice
```

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.

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

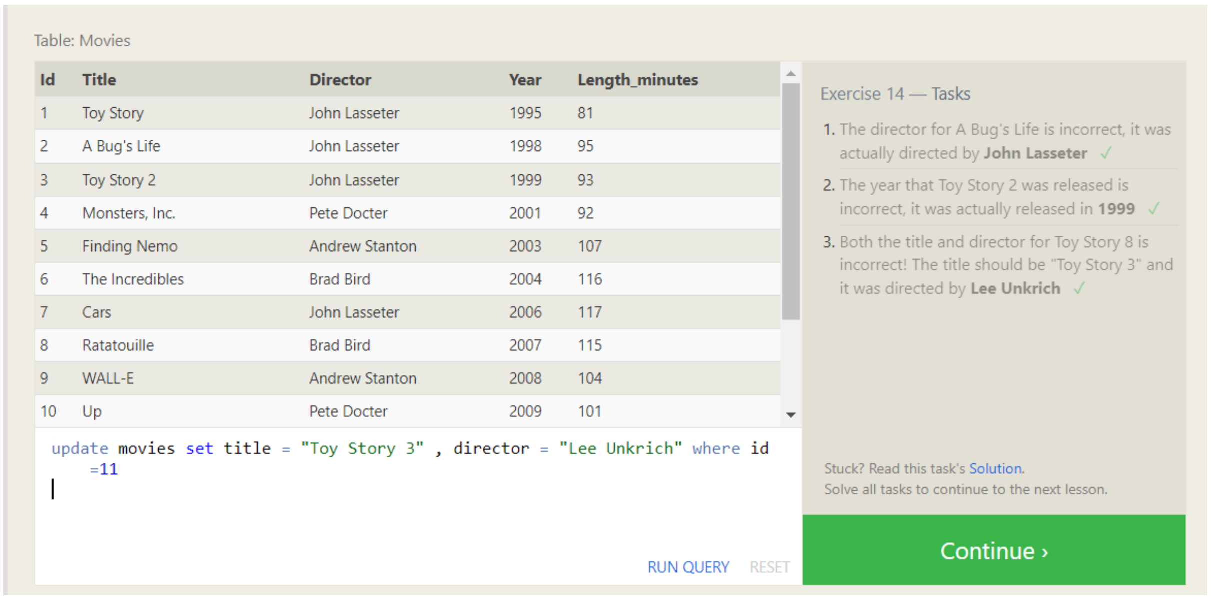


Table: Movies

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

```
update movies set title = "Toy Story 3" , director = "Lee Unkrich" where id
    =11
```

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.

SQL Lesson 15: Deleting rows

1. delete from movies where year < 2005
2. delete from movies where director = "Andrew Stanton"

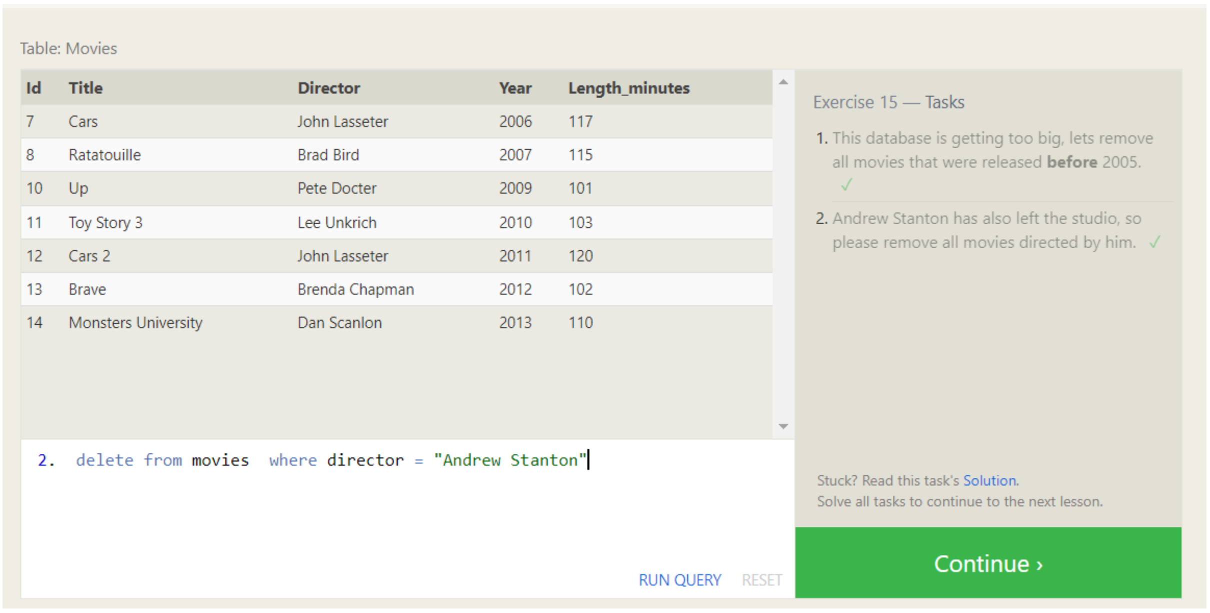


Table: Movies

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

```
2.  delete from movies  where director = "Andrew Stanton"
```

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.

SQL Lesson 16: Creating tables

1. create table database (
   Name varchar(255),
   Version decimal,
   Download_count interger

   );

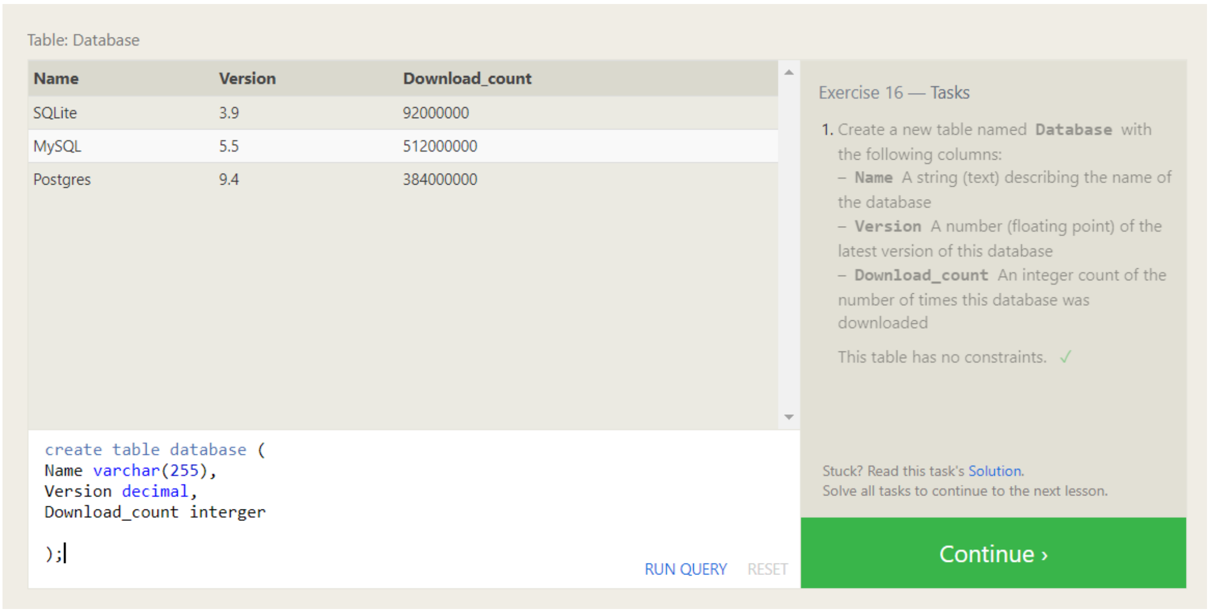


Table: Database

Name	Version	Download_count
SQLite	3.9	92000000
MySQL	5.5	512000000
Postgres	9.4	384000000

```
create table database (
Name varchar(255),
Version decimal,
Download_count interger

);
```

Exercise 16 — Tasks

1. Create a new table named `Database` with the following columns:
   – `Name` A string (text) describing the name of the database
   – `Version` A number (floating point) of the latest version of this database
   – `Download_count` An integer count of the number of times this database was downloaded

   This table has no constraints. ✓

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

SQL Lesson 17: Altering tables

1. alter table movies add column Aspect_ratio float;
2. alter table movies add column Language text default "English"

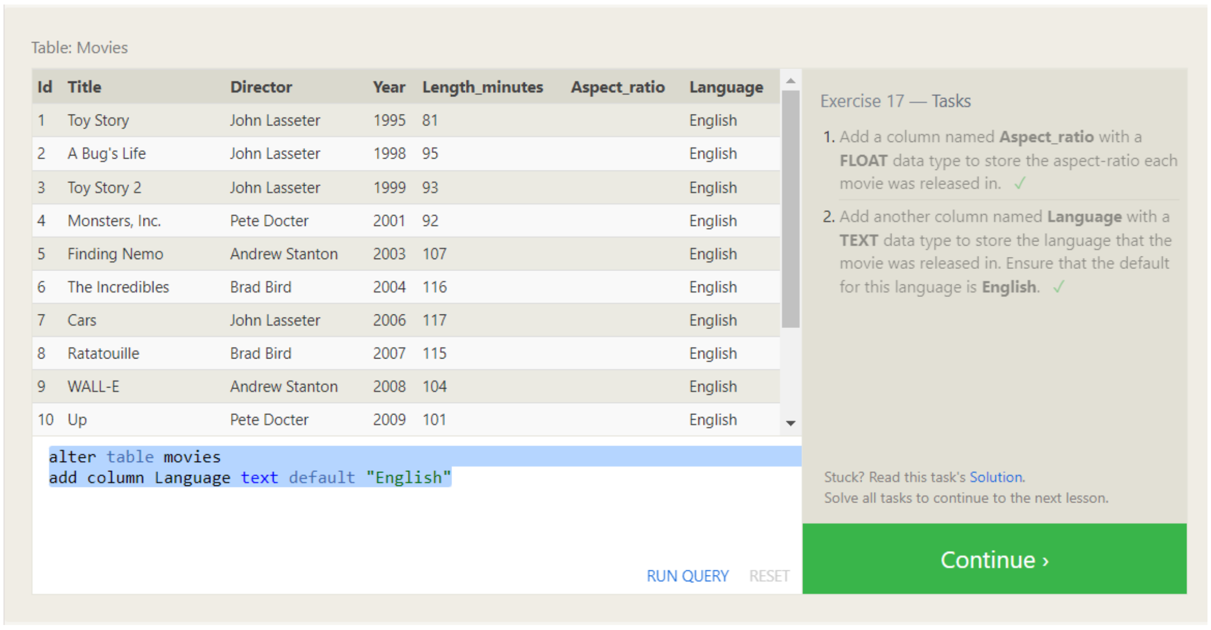


Table: Movies

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

```
alter table movies
add column Language text default "English"
```

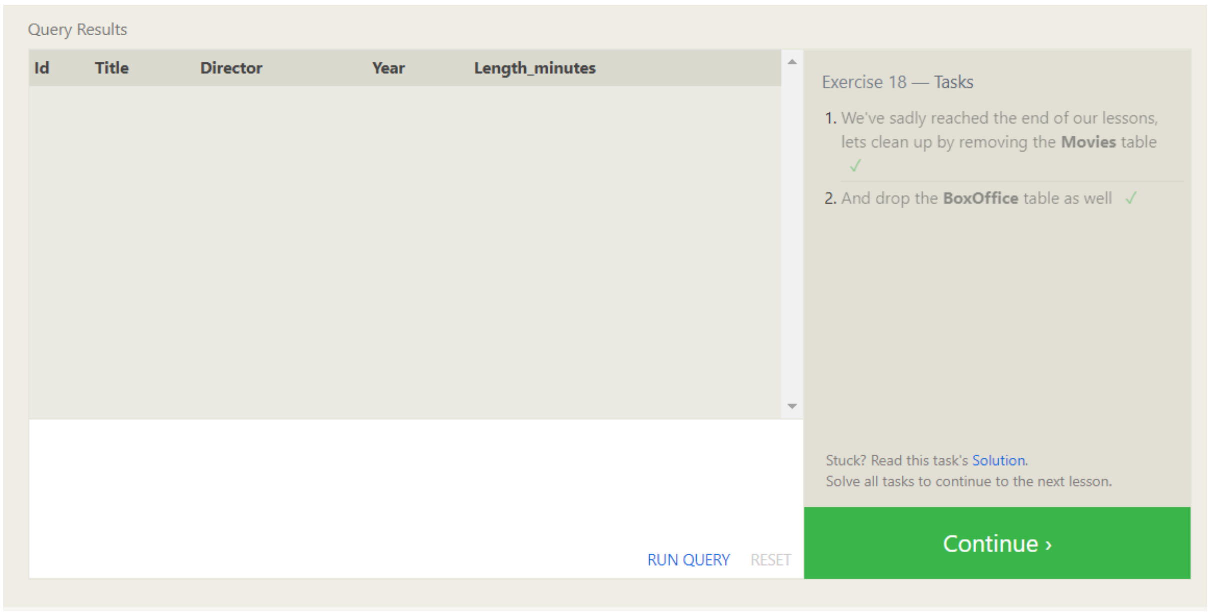
Exercise 17 — Tasks

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.

SQL Lesson 18: Dropping tables

1. drop table movies
2. drop table BoxOffice

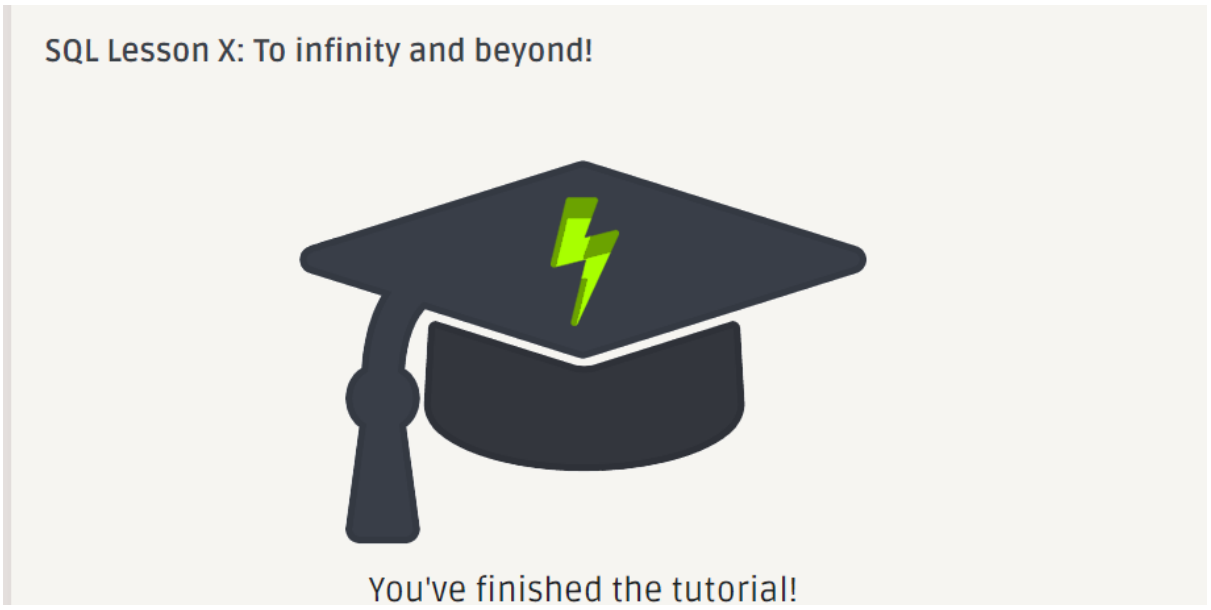

Query Results

Id	Title	Director	Year	Length_minutes

Exercise 18 — Tasks

1. We've sadly reached the end of our lessons, lets clean up by removing the **Movies** table ✓
2. And drop the **BoxOffice** table as well ✓

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


SQL Lesson X: To infinity and beyond!

You've finished the tutorial!