My SQL day 1

SQL Lesson 1: SELECT queries 101

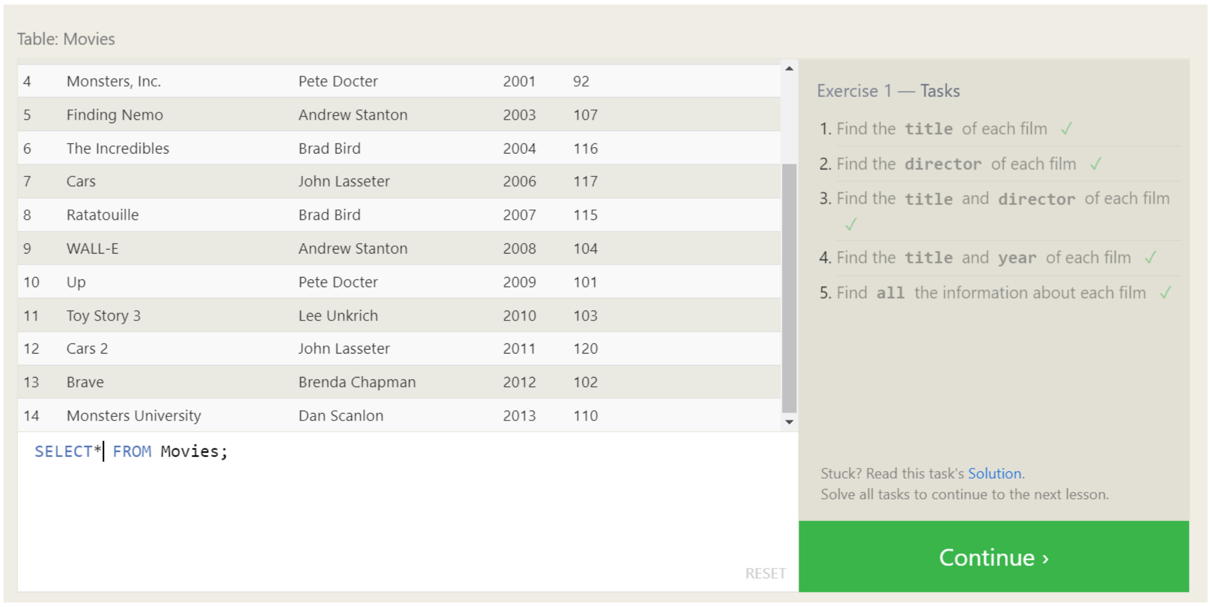


Table: Movies

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
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

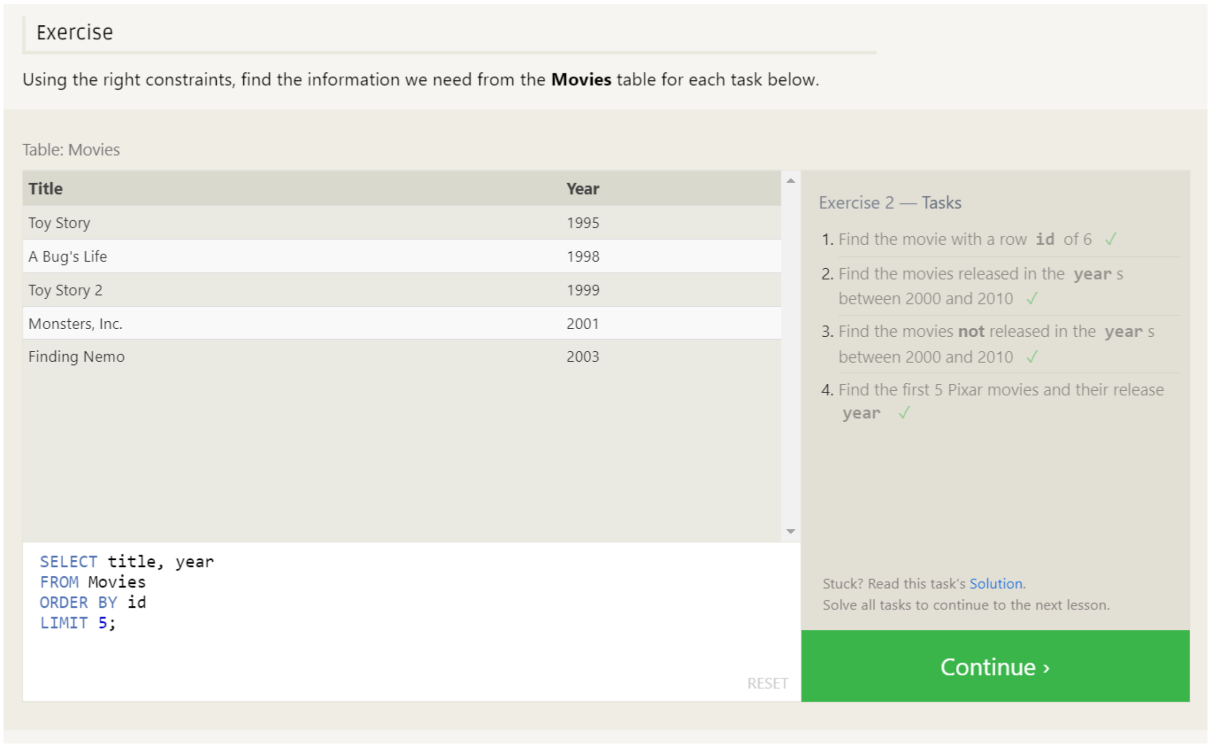
```
SELECT* FROM Movies;
```

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)



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

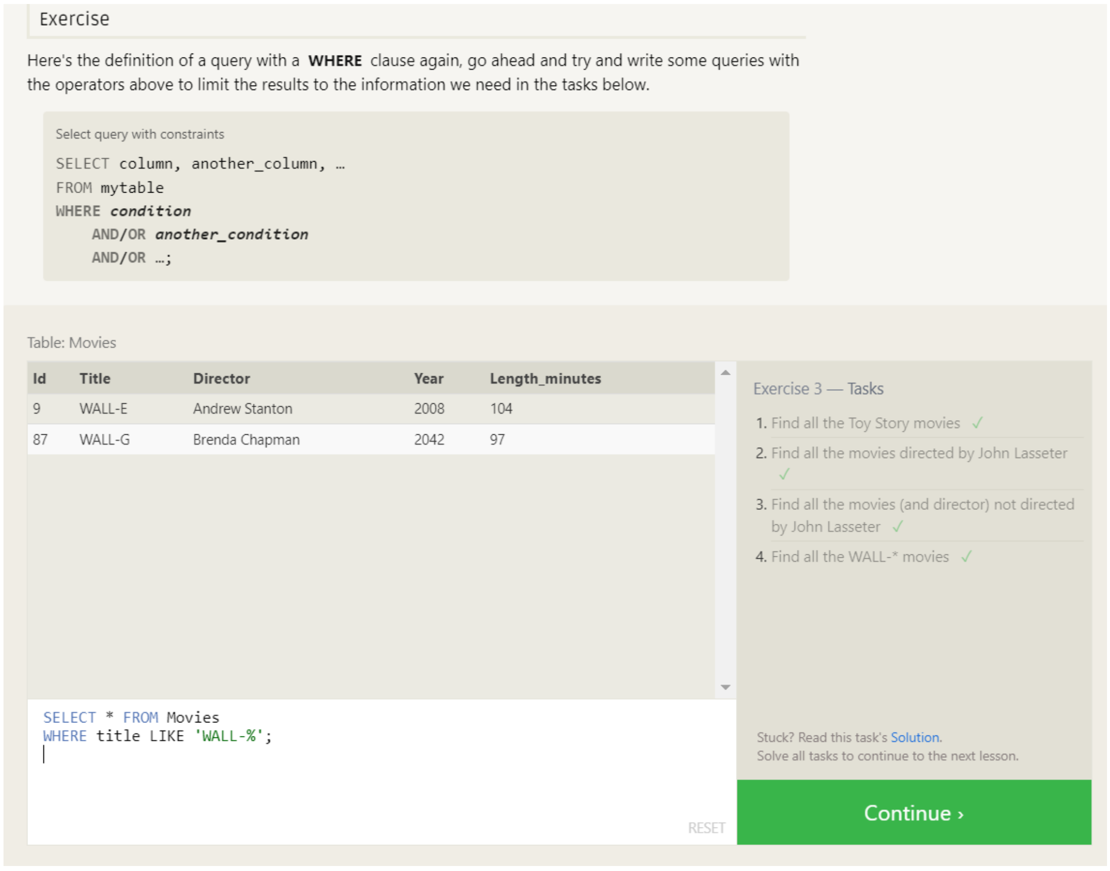
```
SELECT title, year
FROM Movies
ORDER BY id
LIMIT 5;
```

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)



Exercise

Here's the definition of a query with a **WHERE** clause again, go ahead and try and write some queries with the operators above to limit the results to the information we need in the tasks below.

Select query with constraints

```
SELECT column, another_column, …
FROM mytable
WHERE condition
    AND/OR another_condition
    AND/OR …;
```

Table: Movies

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

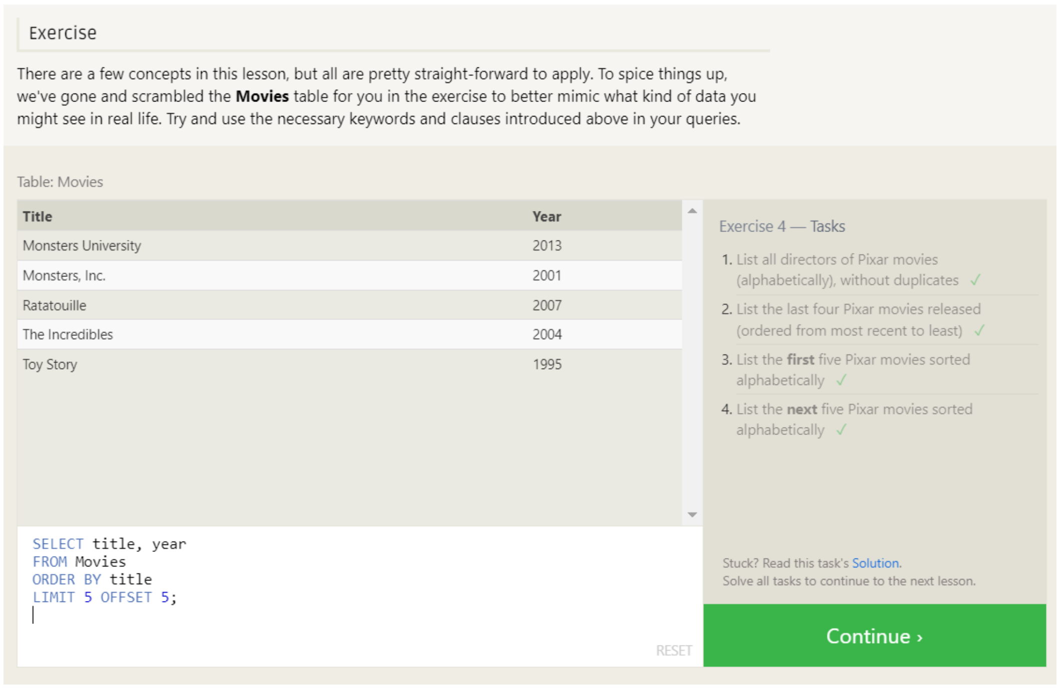
```
SELECT * FROM Movies
WHERE title LIKE 'WALL-%';
```

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



Exercise

There are a few concepts in this lesson, but all are pretty straight-forward to apply. To spice things up, we've gone and scrambled the **Movies** table for you in the exercise to better mimic what kind of data you might see in real life. Try and use the necessary keywords and clauses introduced above in your queries.

Table: Movies

Title	Year
Monsters University	2013
Monsters, Inc.	2001
Ratatouille	2007
The Incredibles	2004
Toy Story	1995

```
SELECT title, year
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

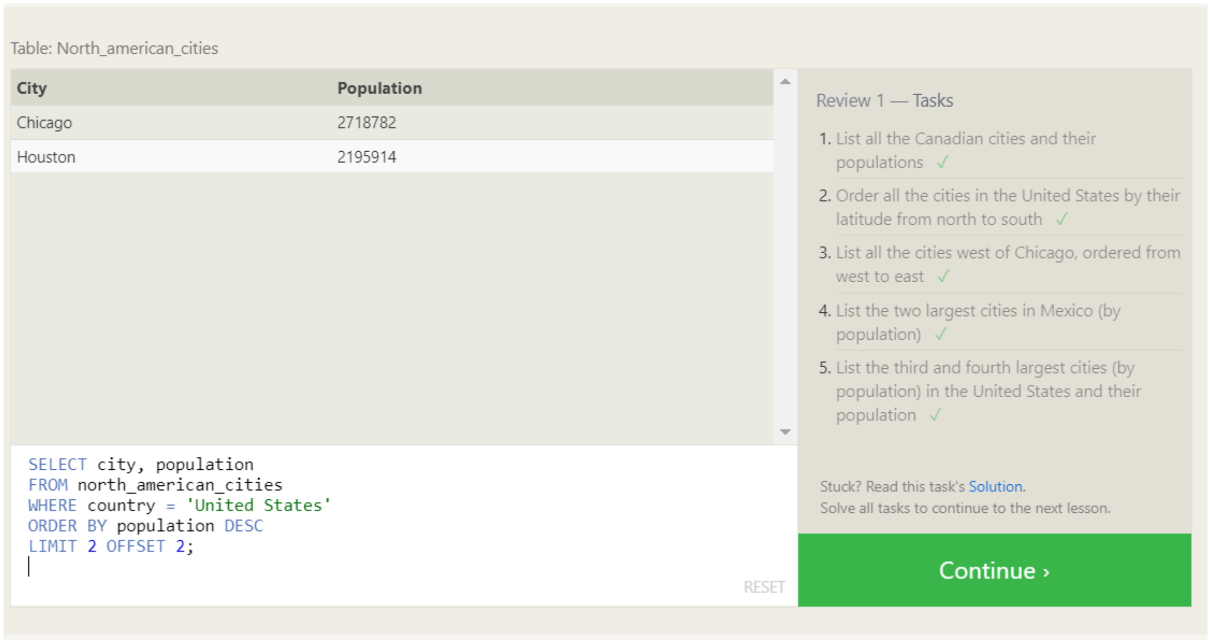


Table: North_american_cities

City	Population
Chicago	2718782
Houston	2195914

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

RESET

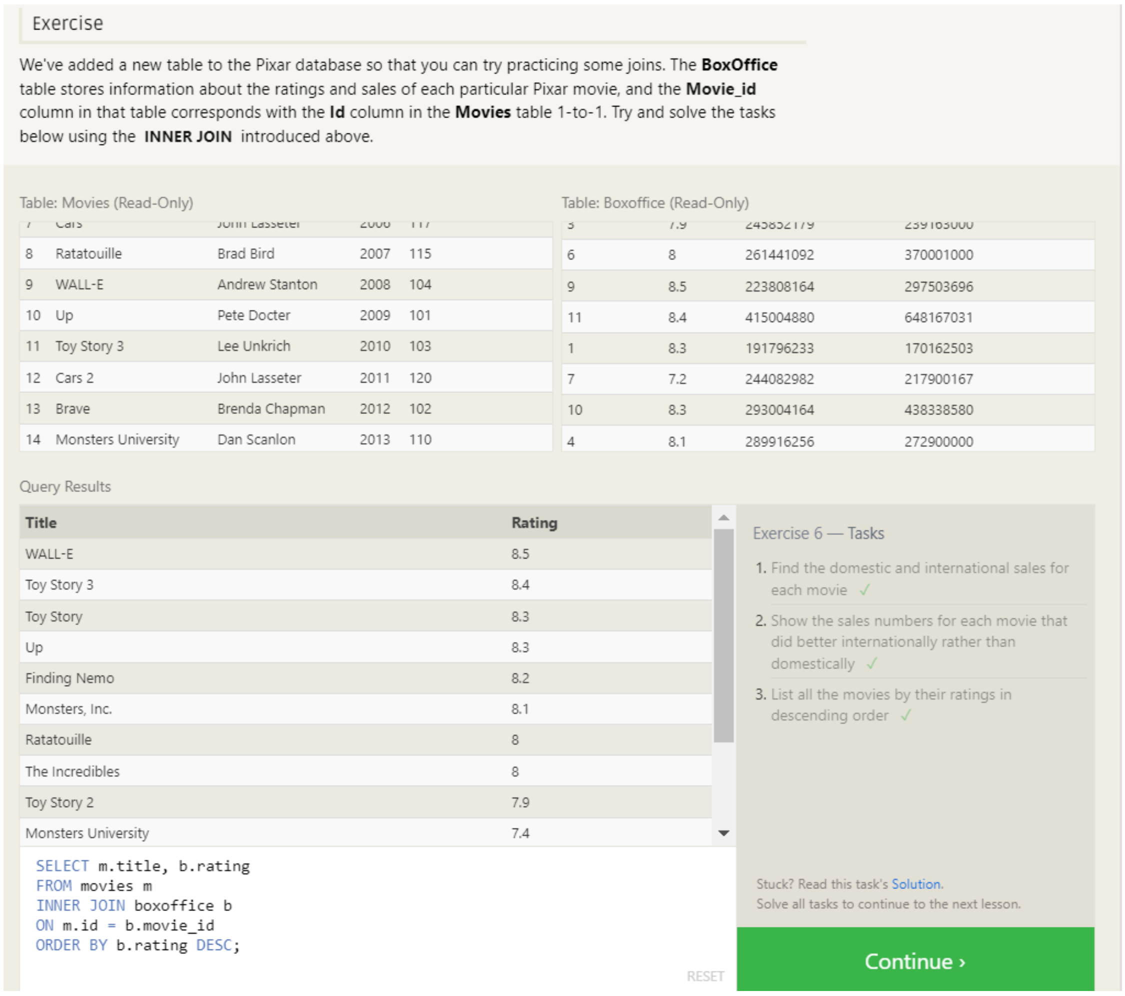
Review 1 — Tasks

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 ›

SQL Lesson 6: Multi-table queries with JOINs



Exercise

We've added a new table to the Pixar database so that you can try practicing some joins. The **BoxOffice** table stores information about the ratings and sales of each particular Pixar movie, and the **Movie_id** column in that table corresponds with the **Id** column in the **Movies** table 1-to-1. Try and solve the tasks below using the **INNER JOIN** introduced above.

Table: Movies (Read-Only)

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
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

Table: Boxoffice (Read-Only)

5	7.9	245852179	239163000
6	8	261441092	370001000
9	8.5	223808164	297503696
11	8.4	415004880	648167031
1	8.3	191796233	170162503
7	7.2	244082982	217900167
10	8.3	293004164	438338580
4	8.1	289916256	272900000

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
INNER JOIN boxoffice b
ON m.id = b.movie_id
ORDER BY b.rating DESC;
```

RESET

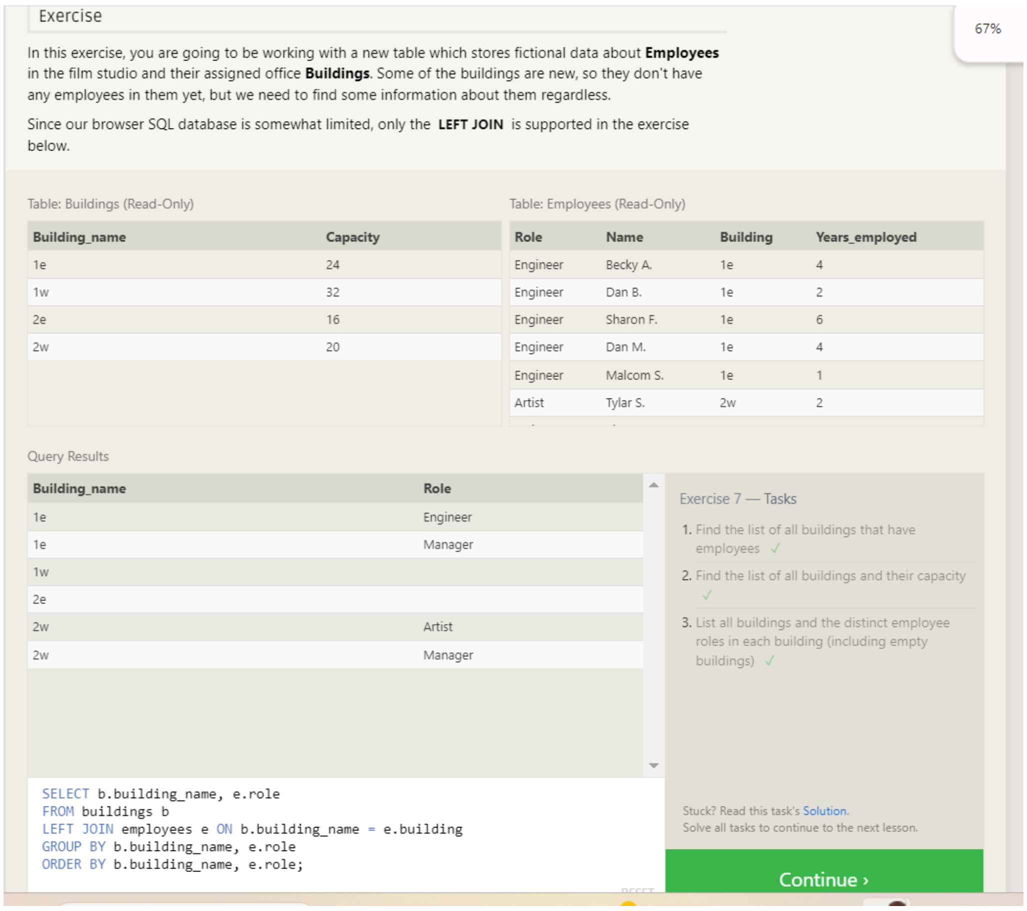
Exercise 6 — Tasks

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



Exercise

67%

In this exercise, you are going to be working with a new table which stores fictional data about **Employees** in the film studio and their assigned office **Buildings**. Some of the buildings are new, so they don't have any employees in them yet, but we need to find some information about them regardless.

Since our browser SQL database is somewhat limited, only the **LEFT JOIN** is supported in the exercise below.

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

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

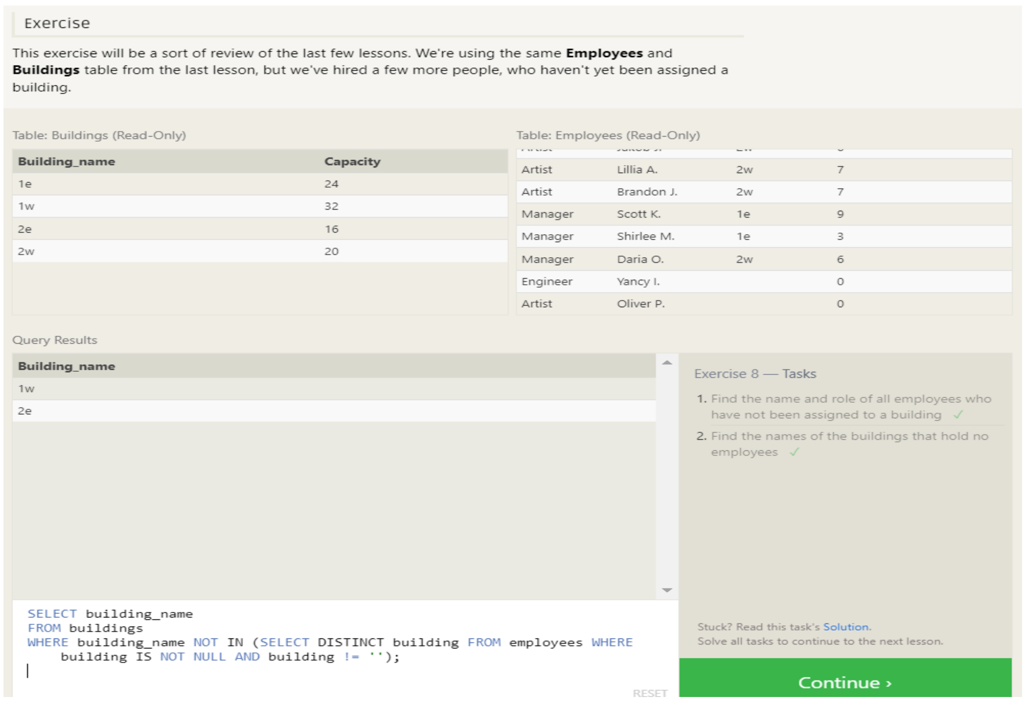
```sql
SELECT b.building_name, e.role
FROM buildings b
LEFT JOIN employees e ON b.building_name = e.building
GROUP BY b.building_name, e.role
ORDER BY b.building_name, e.role;
```

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



Exercise

This exercise will be a sort of review of the last few lessons. We're using the same **Employees** and **Buildings** table from the last lesson, but we've hired a few more people, who haven't yet been assigned a building.

Table: Buildings (Read-Only)

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

Table: Employees (Read-Only)

Artist	Lillia A.	2w	7
Artist	Brandon J.	2w	7
Manager	Scott K.	1e	9
Manager	Shirlee M.	1e	3
Manager	Daria O.	2w	6
Engineer	Yancy I.		0
Artist	Oliver P.		0

Query Results

Building_name
1w
2e

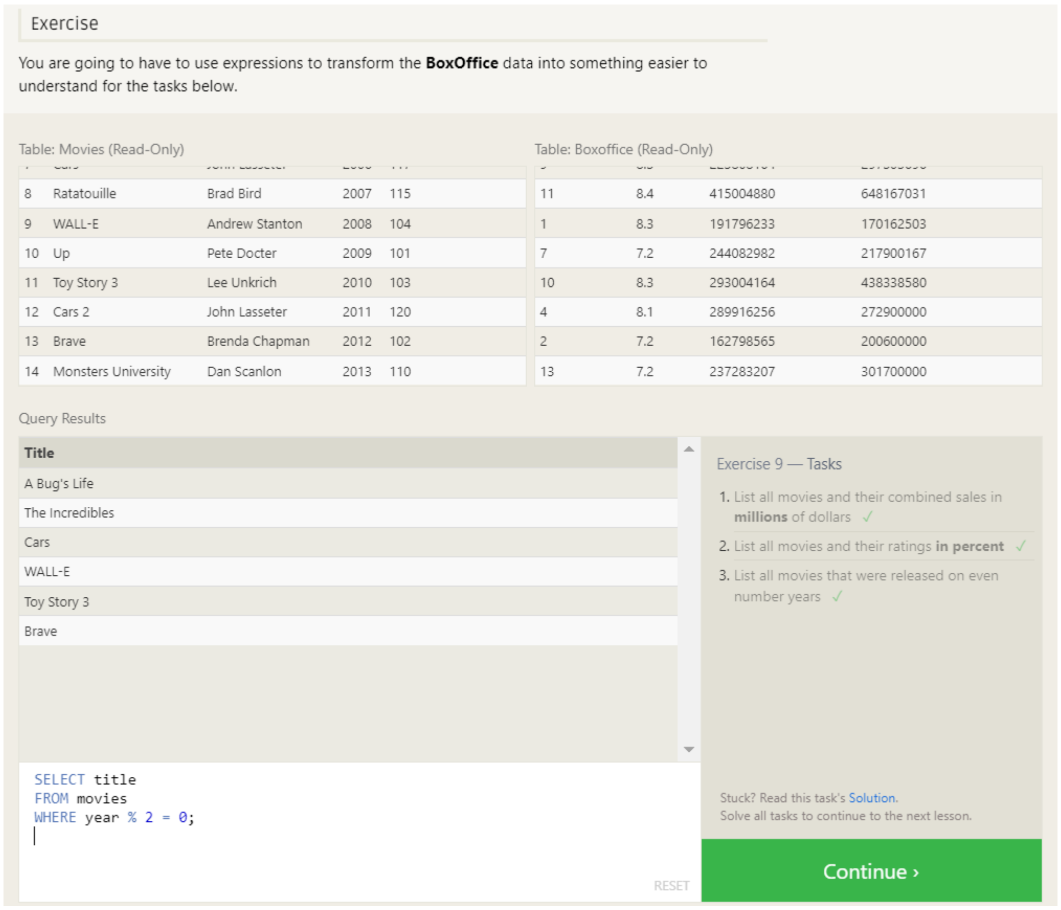
```sql
SELECT building_name
FROM buildings
WHERE building_name NOT IN (SELECT DISTINCT building FROM employees WHERE
    building IS NOT NULL AND building != '');
```

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



Exercise

You are going to have to use expressions to transform the **BoxOffice** data into something easier to understand for the tasks below.

Table: Movies (Read-Only)

8	Ratatouille	Brad Bird	2007	115
9	WALL-E	Andrew Stanton	2008	104
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

Table: Boxoffice (Read-Only)

11	8.4	415004880	648167031
1	8.3	191796233	170162503
7	7.2	244082982	217900167
10	8.3	293004164	438338580
4	8.1	289916256	272900000
2	7.2	162798565	200600000
13	7.2	237283207	301700000

Query Results

Title
A Bug's Life
The Incredibles
Cars
WALL-E
Toy Story 3
Brave

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

RESET

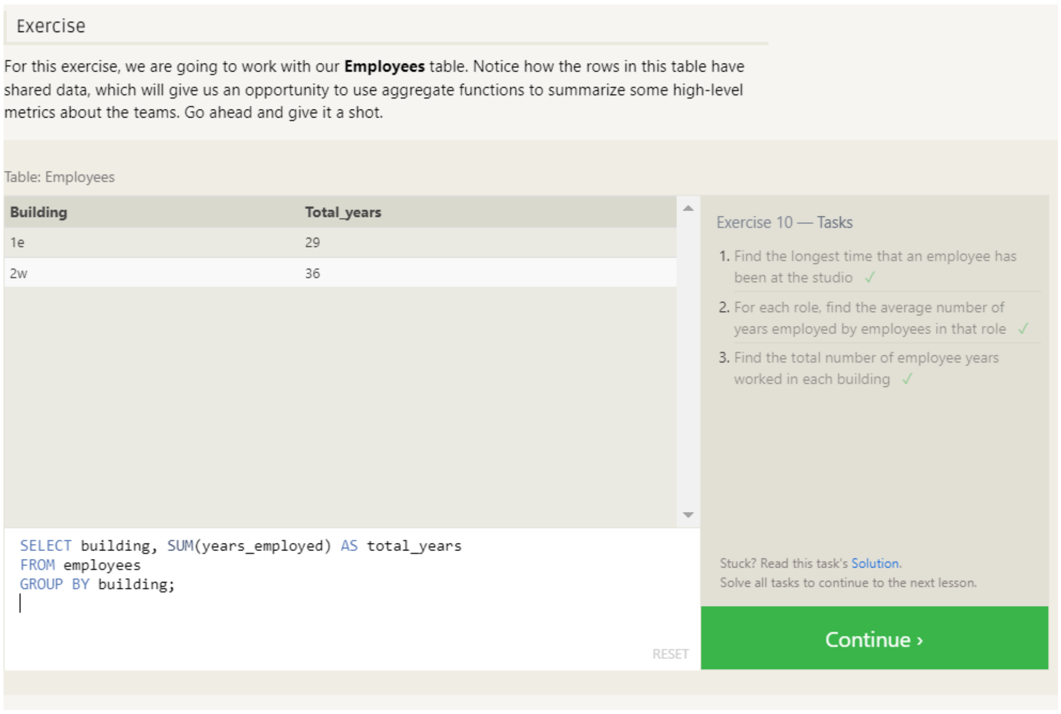
Exercise 9 — Tasks

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 ›

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



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
1e	29
2w	36

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

RESET

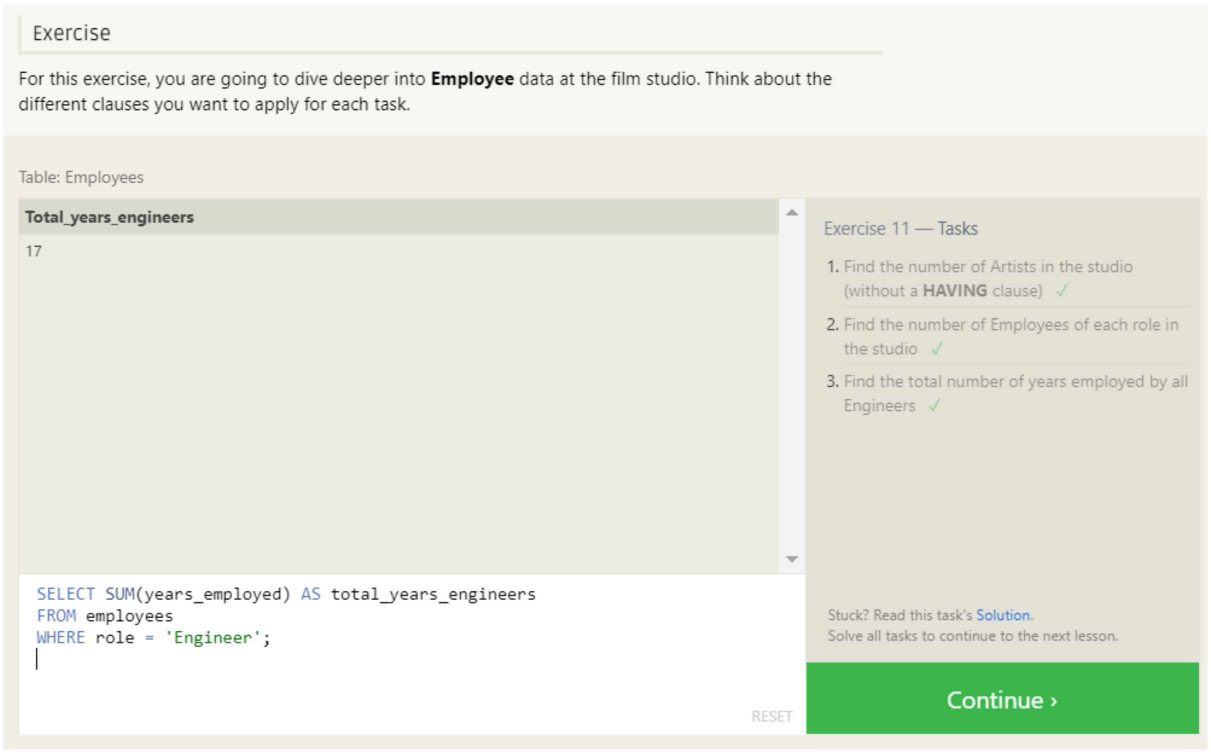
Exercise 10 — Tasks

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 ›

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



Exercise

For this exercise, you are going to dive deeper into **Employee** data at the film studio. Think about the different clauses you want to apply for each task.

Table: Employees

Total_years_engineers
17

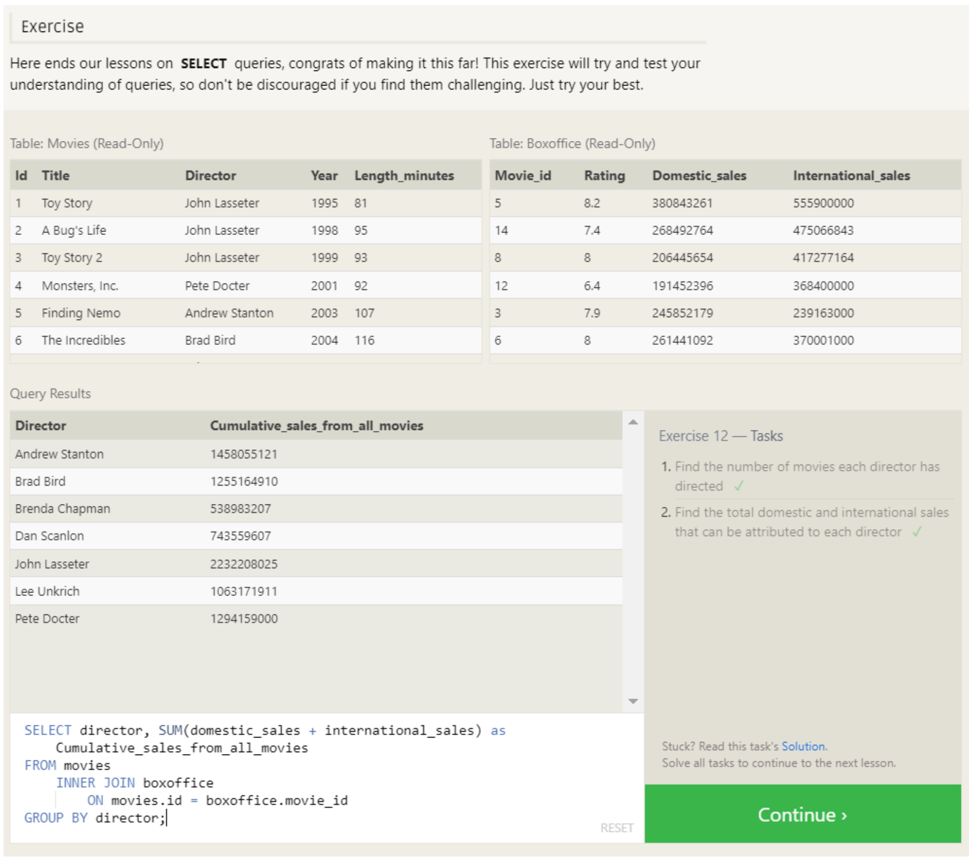
```
SELECT SUM(years_employed) AS total_years_engineers
FROM employees
WHERE role = 'Engineer';
```

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



Exercise

Here ends our lessons on **SELECT** queries, congrats of making it this far! This exercise will try and test 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	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

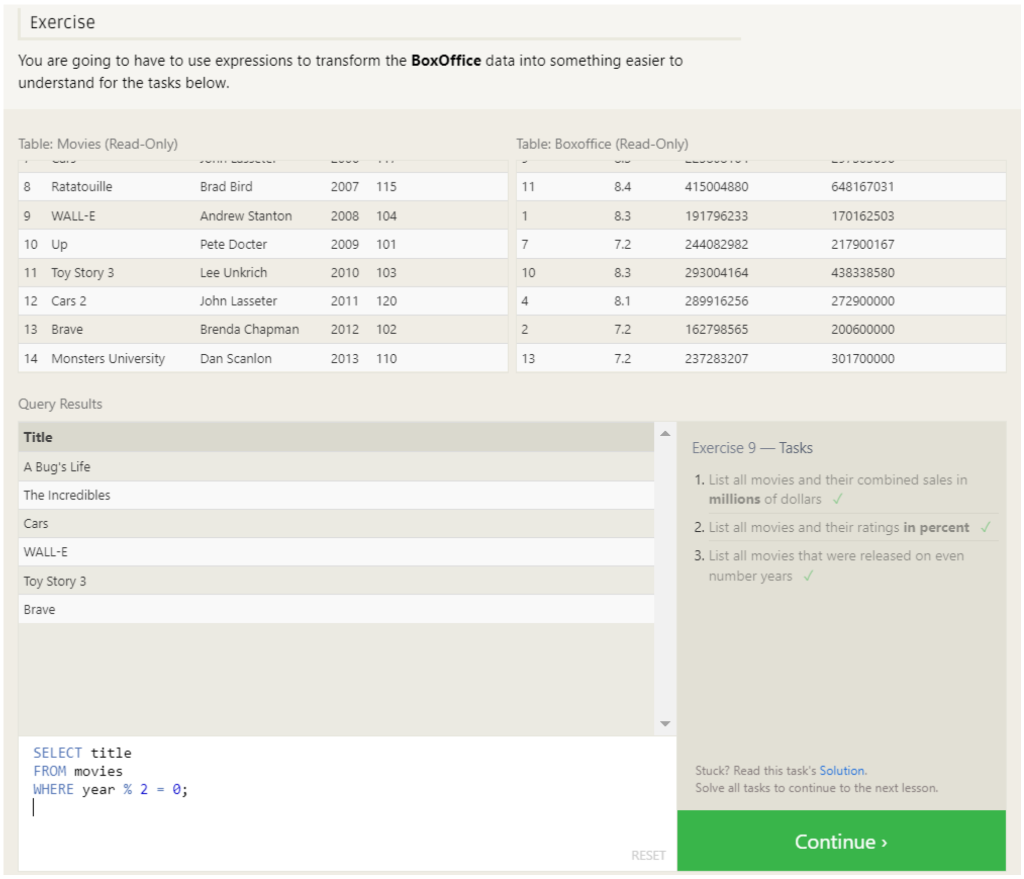
```
SELECT director, SUM(domestic_sales + international_sales) as
    Cumulative_sales_from_all_movies
FROM movies
    INNER JOIN boxoffice
        ON movies.id = boxoffice.movie_id
GROUP BY director;
```

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



Exercise

You are going to have to use expressions to transform the **BoxOffice** data into something easier to understand for the tasks below.

Table: Movies (Read-Only)

8	Ratatouille	Brad Bird	2007	115
9	WALL-E	Andrew Stanton	2008	104
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

Table: Boxoffice (Read-Only)

11	8.4	415004880	648167031
1	8.3	191796233	170162503
7	7.2	244082982	217900167
10	8.3	293004164	438338580
4	8.1	289916256	272900000
2	7.2	162798565	200600000
13	7.2	237283207	301700000

Query Results

Title
A Bug's Life
The Incredibles
Cars
WALL-E
Toy Story 3
Brave

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

RESET

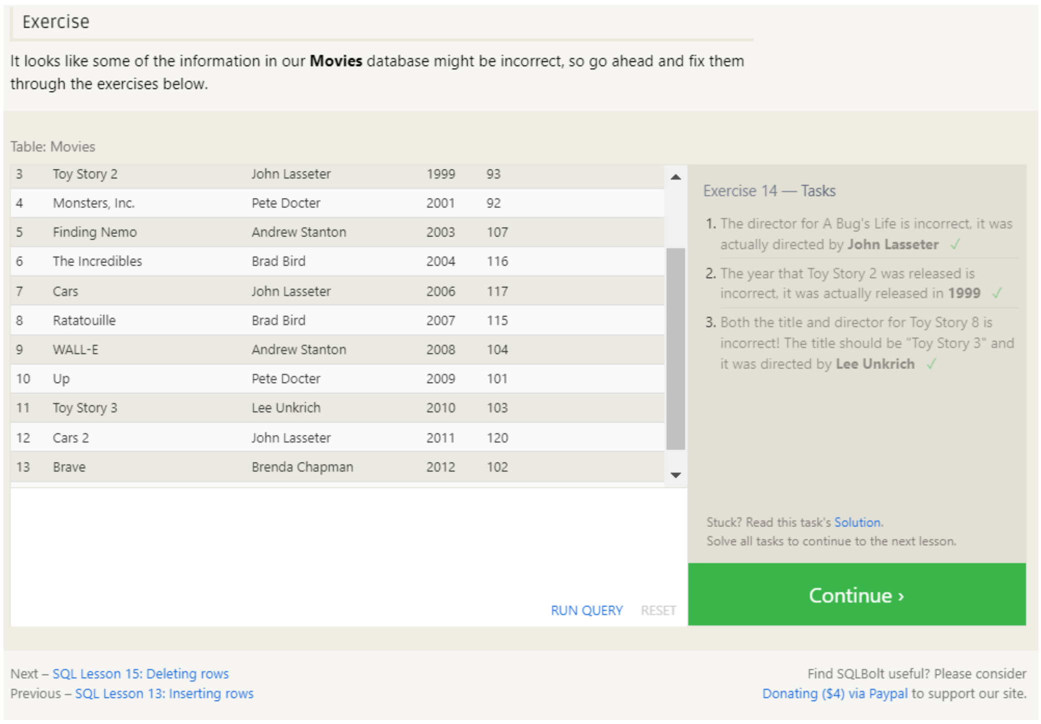
Exercise 9 — Tasks

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 ›

SQL Lesson 14: Updating rows



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

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
11	Toy Story 3	Lee Unkrich	2010	103
12	Cars 2	John Lasseter	2011	120
13	Brave	Brenda Chapman	2012	102

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** ✓

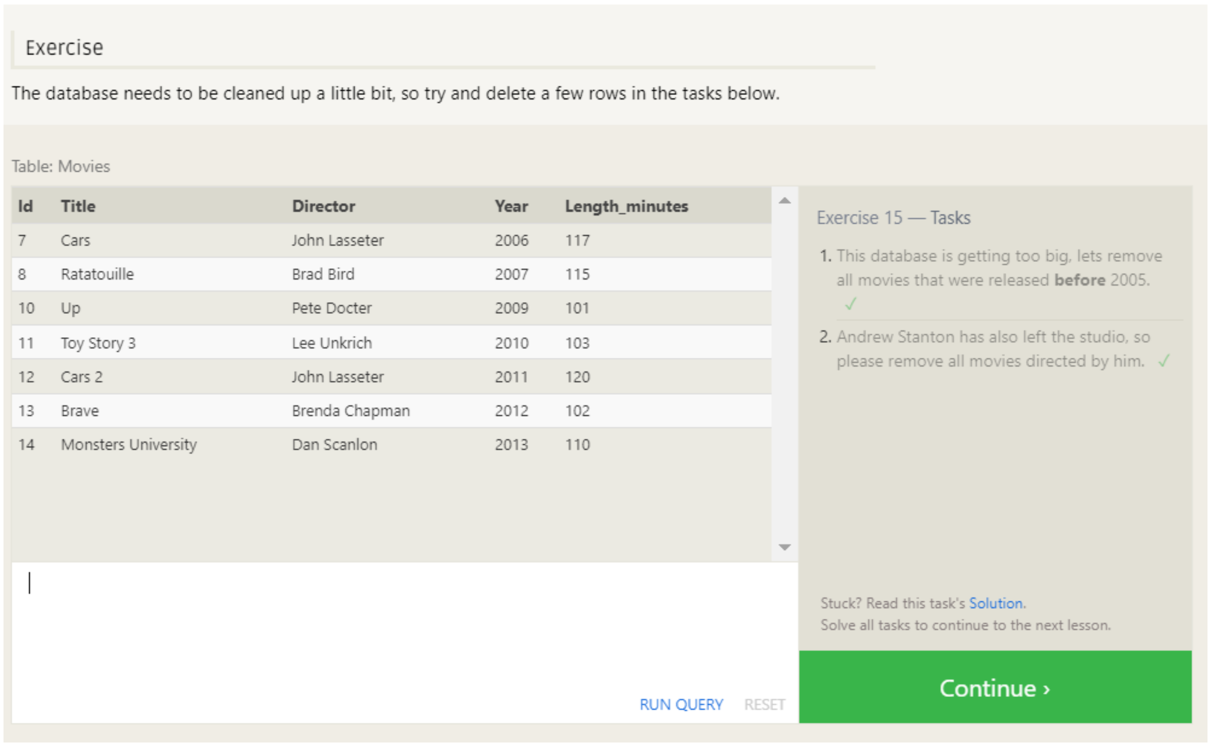
Stuck? Read this task's Solution.
Solve all tasks to continue to the next lesson.

Continue ›

Next – SQL Lesson 15: Deleting rows
Previous – SQL Lesson 13: Inserting rows

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

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

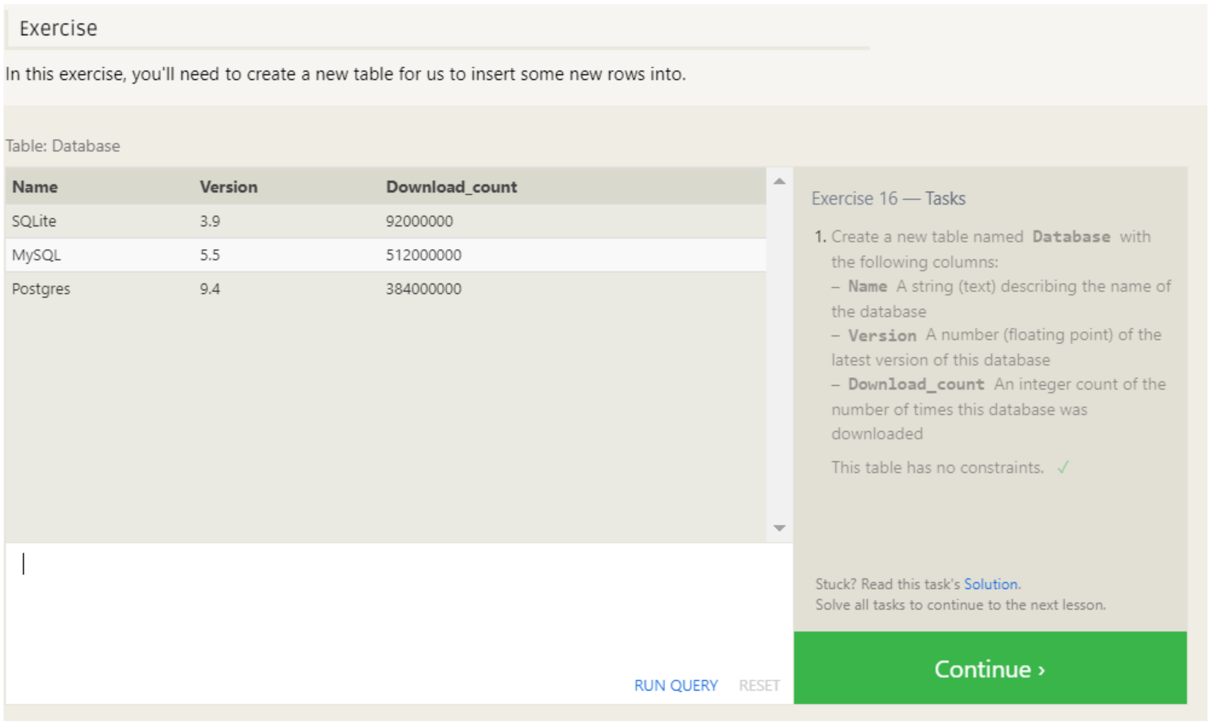
Id	Title	Director	Year	Length_minutes
7	Cars	John Lasseter	2006	117
8	Ratatouille	Brad Bird	2007	115
10	Up	Pete Docter	2009	101
11	Toy Story 3	Lee Unkrich	2010	103
12	Cars 2	John Lasseter	2011	120
13	Brave	Brenda Chapman	2012	102
14	Monsters University	Dan Scanlon	2013	110

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



Exercise

In this exercise, you'll need to create a new table for us to insert some new rows into.

Table: Database

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

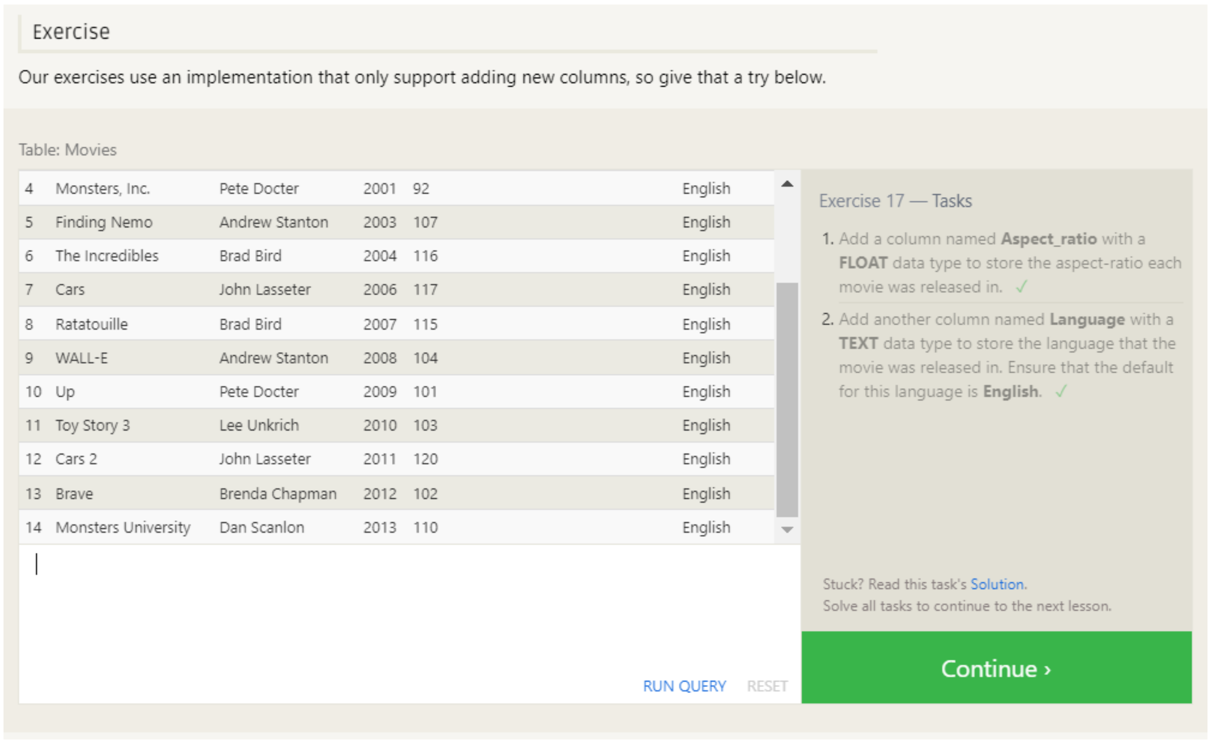
Exercise 16 — Tasks

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



Exercise

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

Table: Movies

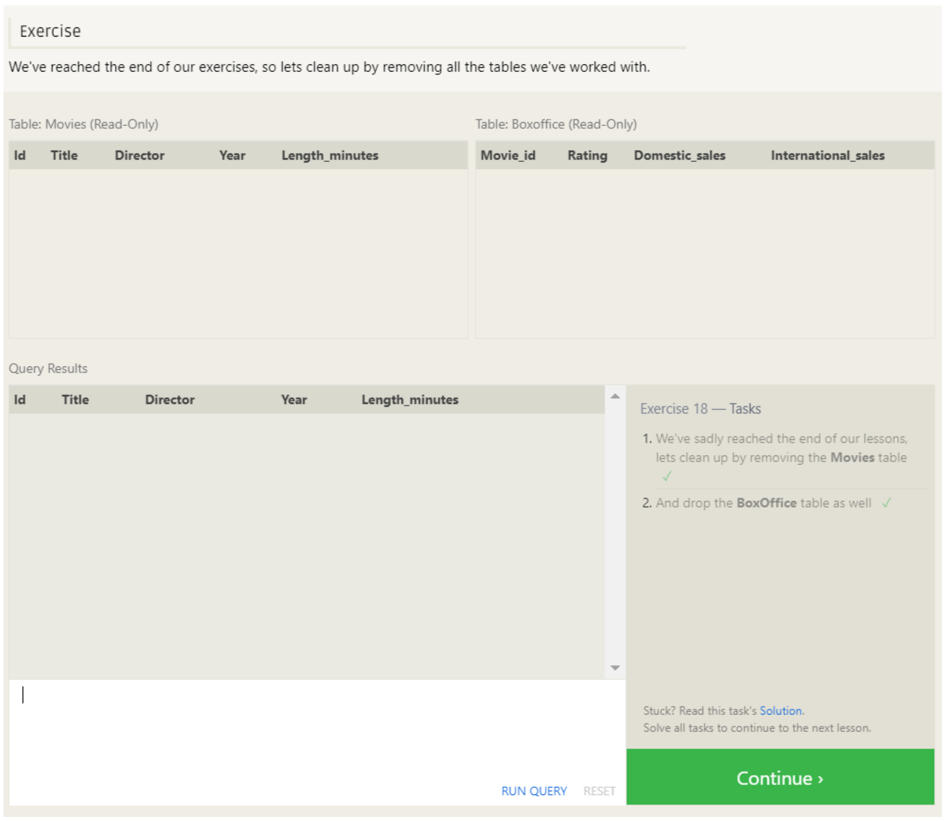
4	Monsters, Inc.	Pete Docter	2001	92	English
5	Finding Nemo	Andrew Stanton	2003	107	English
6	The Incredibles	Brad Bird	2004	116	English
7	Cars	John Lasseter	2006	117	English
8	Ratatouille	Brad Bird	2007	115	English
9	WALL-E	Andrew Stanton	2008	104	English
10	Up	Pete Docter	2009	101	English
11	Toy Story 3	Lee Unkrich	2010	103	English
12	Cars 2	John Lasseter	2011	120	English
13	Brave	Brenda Chapman	2012	102	English
14	Monsters University	Dan Scanlon	2013	110	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



Exercise

We've reached the end of our exercises, so lets clean up by removing all the tables we've worked with.

Table: Movies (Read-Only)

Id	Title	Director	Year	Length_minutes

Table: Boxoffice (Read-Only)

Movie_id	Rating	Domestic_sales	International_sales

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.