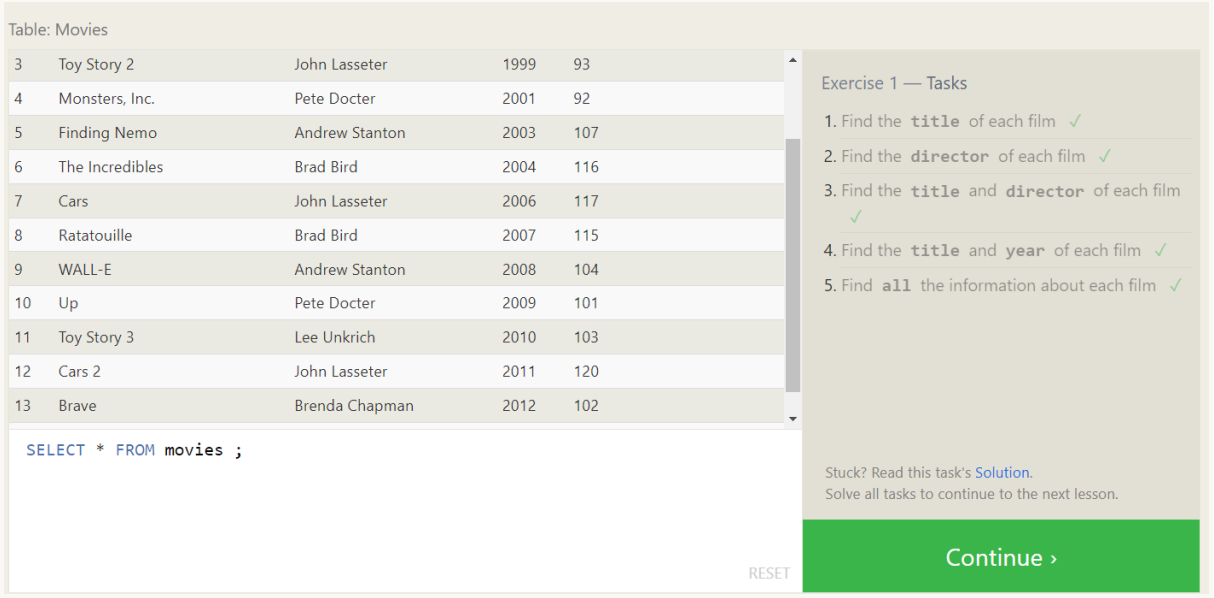
Day -1 : Database - MySQL

SQL Lesson 1: SELECT queries

1) Find the title of each film : SELECT title FROM movies ;
2) Find the director of each film : SELECT director FROM movies ;
3) Find the title and director of each film : SELECT title,director FROM movies ;
4) Find the title and year of each film : SELECT title,year FROM movies ;
5) Find all the information about each film : SELECT * FROM movies ;



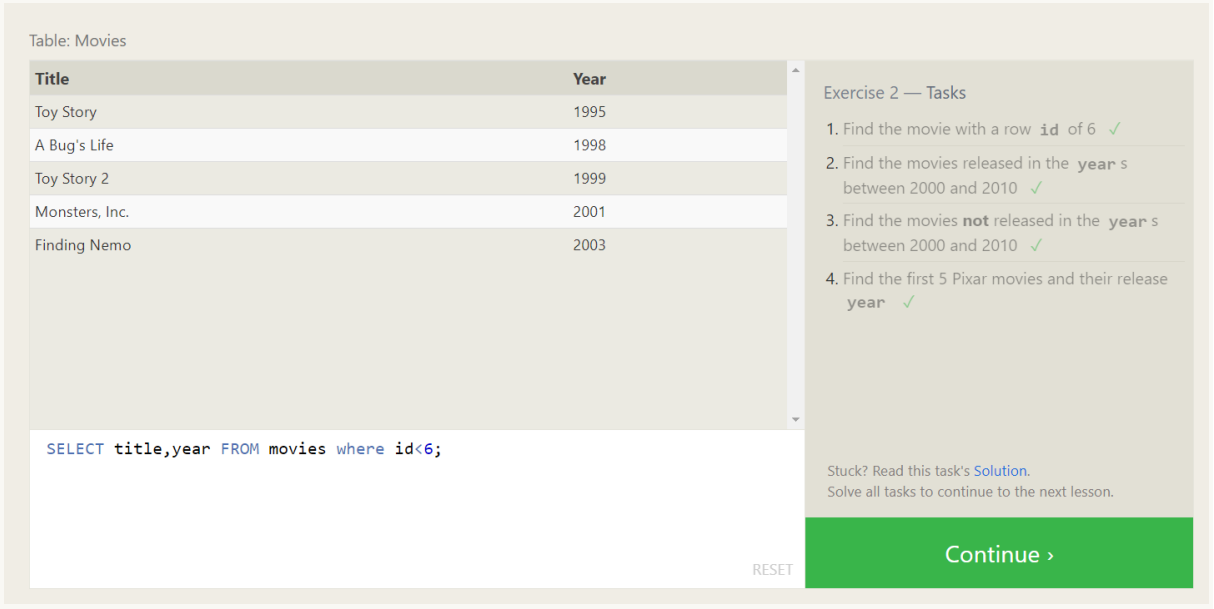
SQL Lesson 2: Queries with constraints

1)Find the movie with a row **id** of 6 : SELECT * FROM movies where id=6;

2)Find the movies released in the **year**s between 2000 and 2010 : SELECT * FROM movies where year between 2000 and 2010;

3)Find the movies **not** released in the **year**s between 2000 and 2010 : SELECT * FROM movies where year not between 2000 and 2010;

4)Find the first 5 Pixar movies and their release **year** : `SELECT title,year FROM movies where id<6;`



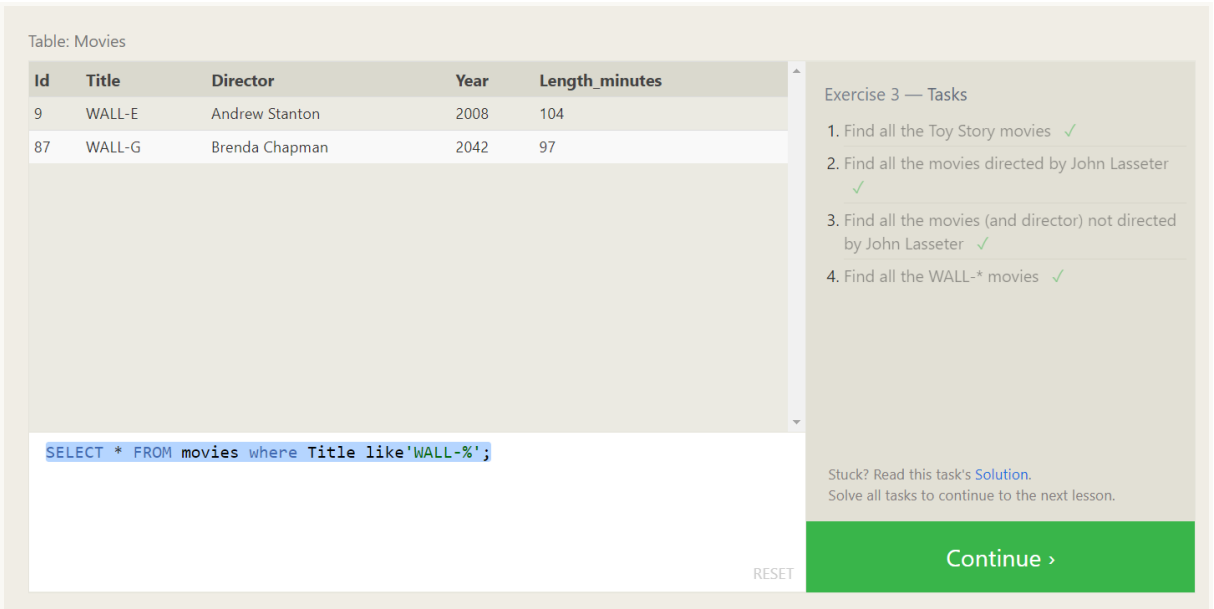
SQL Lesson 3: Queries with constraints

1) Find all the Toy Story movies: SELECT title FROM movies where title like 'TOY STORY%';

2)Find all the movies directed by John Lasseter : SELECT * FROM movies where Director like 'John Lasseter';

3)Find all the movies (and director) not directed by John Lasseter : SELECT * FROM movies where Director not like 'John Lasseter';

4)Find all the WALL-* movies : SELECT * FROM movies where Title like 'WALL-%';



SQL Lesson 4: Filtering and sorting Query results

1) List all directors of Pixar movies (alphabetically), without duplicates : SELECT distinct director FROM movies order by director;
2)List the last four Pixar movies released (ordered from most recent to least) : SELECT title,year FROM movies order by year desc LIMIT 4;
3)List the **first** five Pixar movies sorted alphabetically : SELECT * FROM movies order by title asc LIMIT 5;
4)List the **next** five Pixar movies sorted alphabetically : SELECT * FROM movies order by title asc LIMIT 5 offset 5;

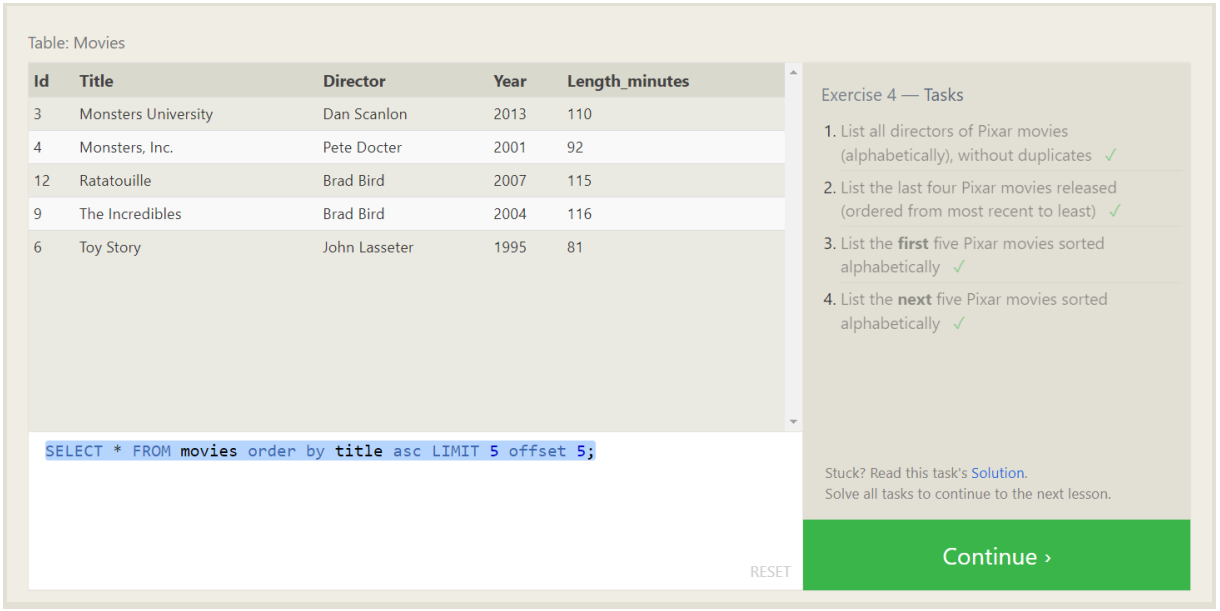


Table: Movies

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

```
SELECT * FROM movies order by title asc LIMIT 5 offset 5;
```

RESET

Exercise 4 — Tasks

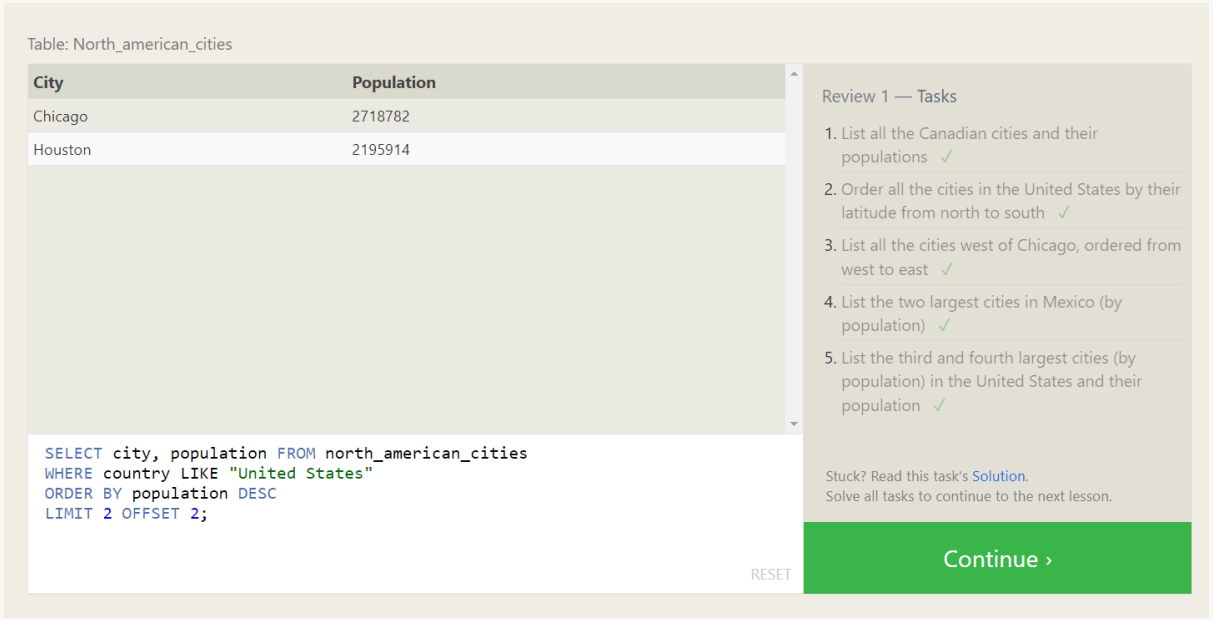
1. List all directors of Pixar movies (alphabetically), without duplicates ✓
2. List the last four Pixar movies released (ordered from most recent to least) ✓
3. List the **first** five Pixar movies sorted alphabetically ✓
4. List the **next** five Pixar movies sorted alphabetically ✓

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

Continue ›

SQL Review : Simple SELECT Queries

1) List all the Canadian cities and their populations: SELECT * FROM north_american_cities where Country='Canada';
2) Order all the cities in the United States by their latitude from north to south :SELECT * FROM north_american_cities where country='United States' order by latitude desc;
3)List all the cities west of Chicago, ordered from west to east : SELECT city, longitude FROM north_american_cities WHERE longitude < -87.629798 ORDER BY longitude ASC;
4)List the third and fourth largest cities (by population) in the United States and their population: SELECT city, population FROM north_american_cities WHERE country LIKE "United States" ORDER BY population DESC LIMIT 2 OFFSET 2;

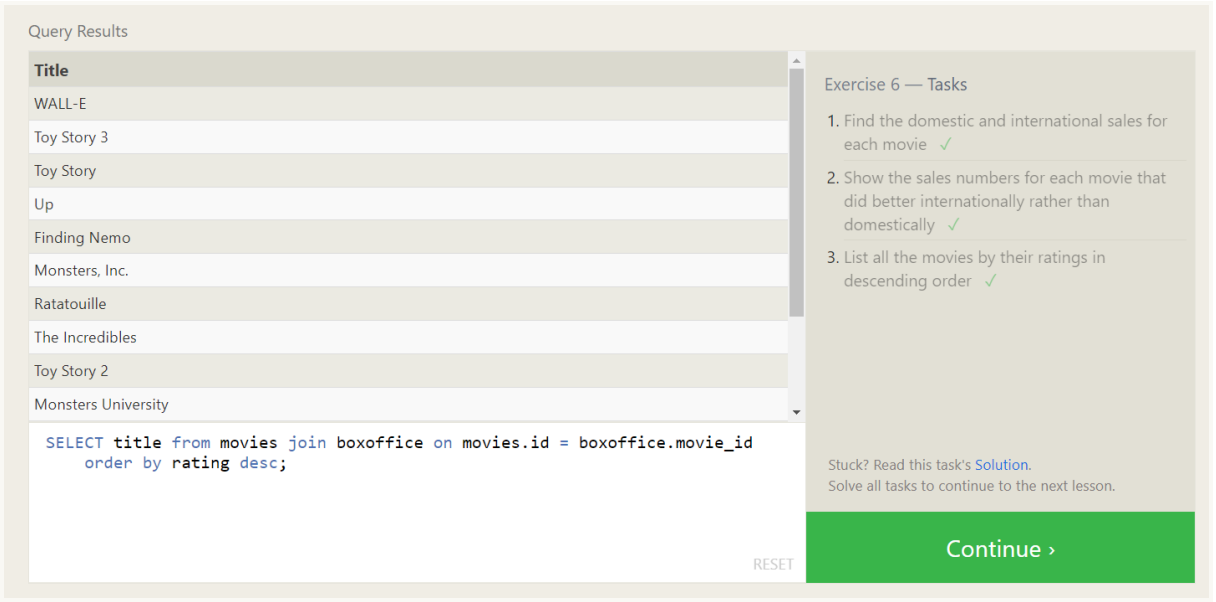


SQL Lesson 6: Multi-table queries with JOINs

1) Find the domestic and international sales for each movie: SELECT title,domestic_sales,international_sales FROM movies join boxoffice on movies.id = boxoffice.movie_id;

2)Show the sales numbers for each movie that did better internationally rather than domestically : SELECT title, domestic_sales, international_sales FROM movies JOIN boxoffice ON movies.id = boxoffice.movie_id WHERE international_sales > domestic_sales;

3)List all the movies by their ratings in descending order : SELECT title from movies join boxoffice on movies.id = boxoffice.movie_id order by rating desc;

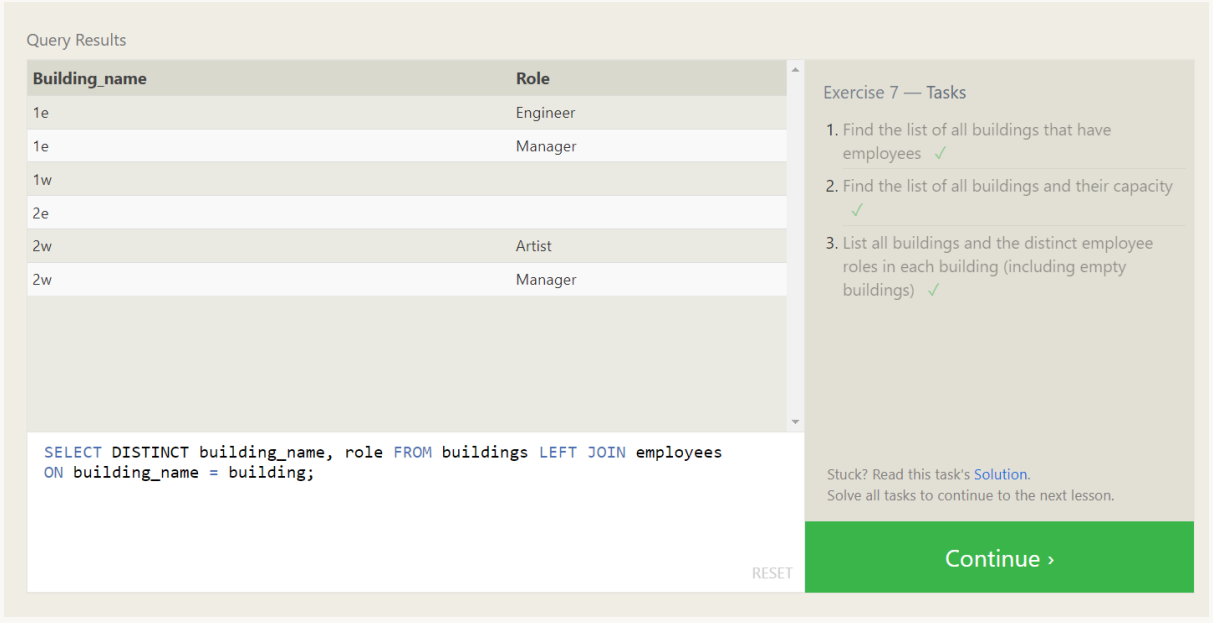


SQL Lesson 7: OUTER JOINs

1) Find the list of all buildings that have employees : SELECT DISTINCT building FROM employees;

2) Find the list of all buildings and their capacity : SELECT * FROM Buildings;

3)List all buildings and the distinct employee roles in each building (including empty buildings) : SELECT DISTINCT building_name, role FROM buildings LEFT JOIN employees ON building_name = building;



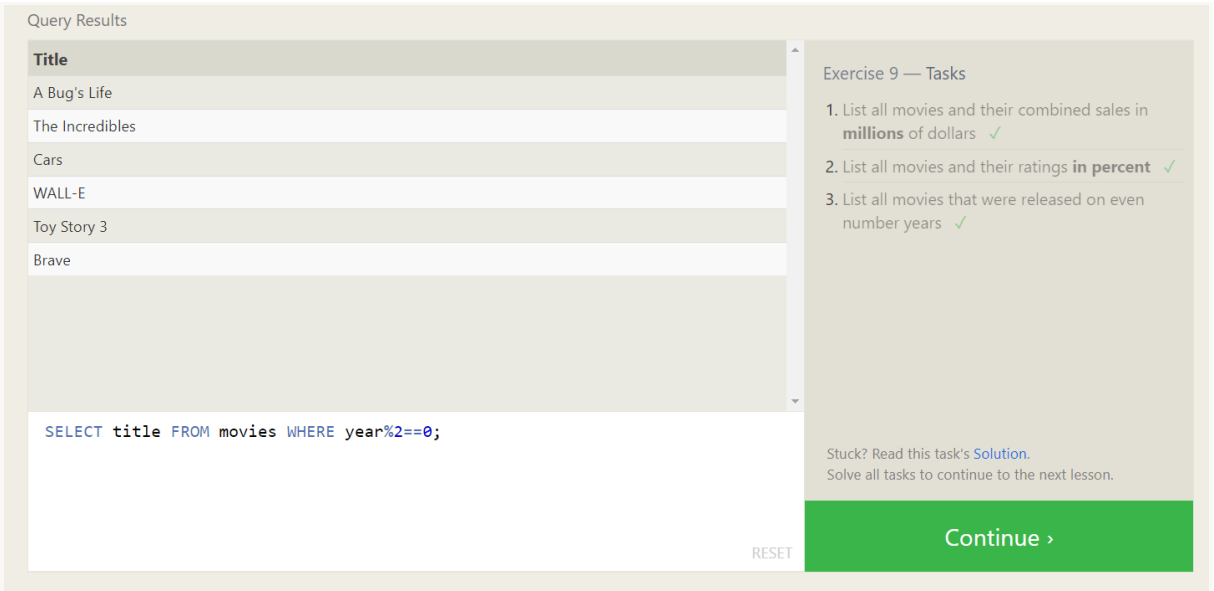
SQL Lesson 8: A short note on NULLs

1) Find the name and role of all employees who have not been assigned to a building: SELECT name, role FROM employees WHERE building IS NULL;

2)Find the names of the buildings that hold no employees : SELECT DISTINCT building_name FROM buildings LEFT JOIN employees ON building_name = building WHERE role IS NULL;

SQL Lesson 9: Queries with expressions

1) List all movies and their combined sales in **millions** of dollars : SELECT title, (domestic_sales + international_sales) / 1000000 AS gross_sales_millions FROM movies JOIN boxoffice ON movies.id = boxoffice.movie_id;
2)List all movies and their ratings **in percent** : SELECT title, Rating*10 AS Rating_Percentage FROM movies JOIN boxoffice ON movies.id = boxoffice.movie_id;
3)List all movies that were released on even number years : SELECT title FROM movies WHERE year%2==0;



SQL Lesson 10: Queries with aggregates

1) Find the longest time that an employee has been at the studio : SELECT MAX(years_employed) as Max_years_employed FROM employees;

2)For each role, find the average number of years employed by employees in that role : SELECT role, AVG(years_employed) as Average_years_employed FROM employees GROUP BY role;

3)Find the total number of employee years worked in each building :SELECT building, SUM(years_employed) as Total_years_employed FROM employees GROUP BY building;

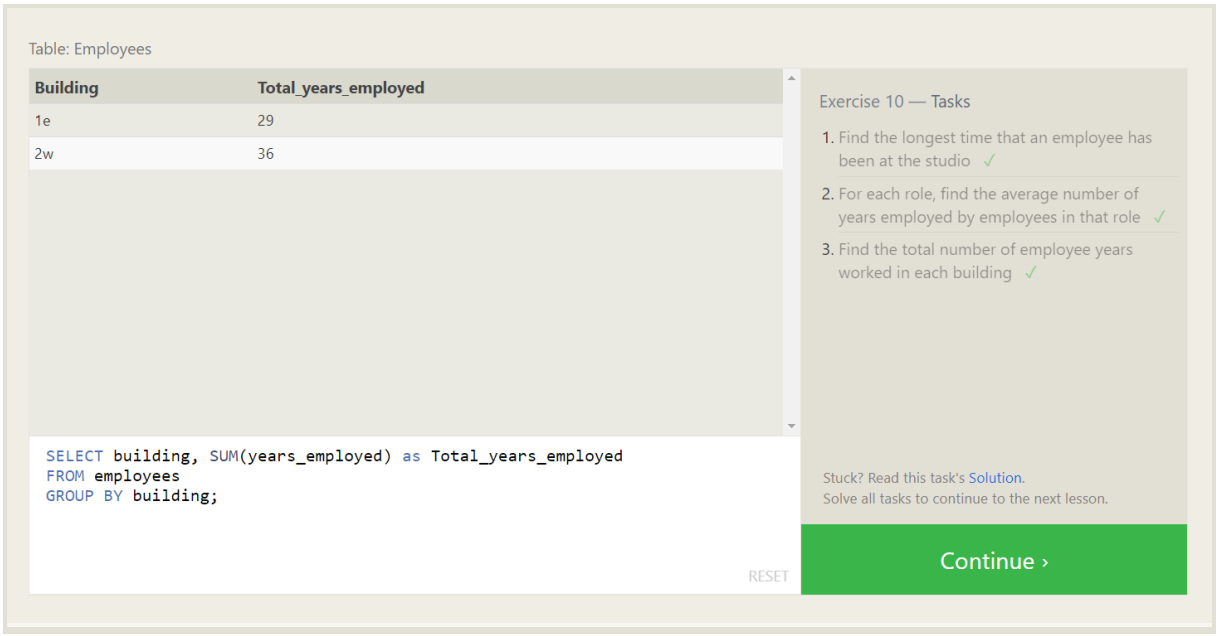


Table: Employees

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.

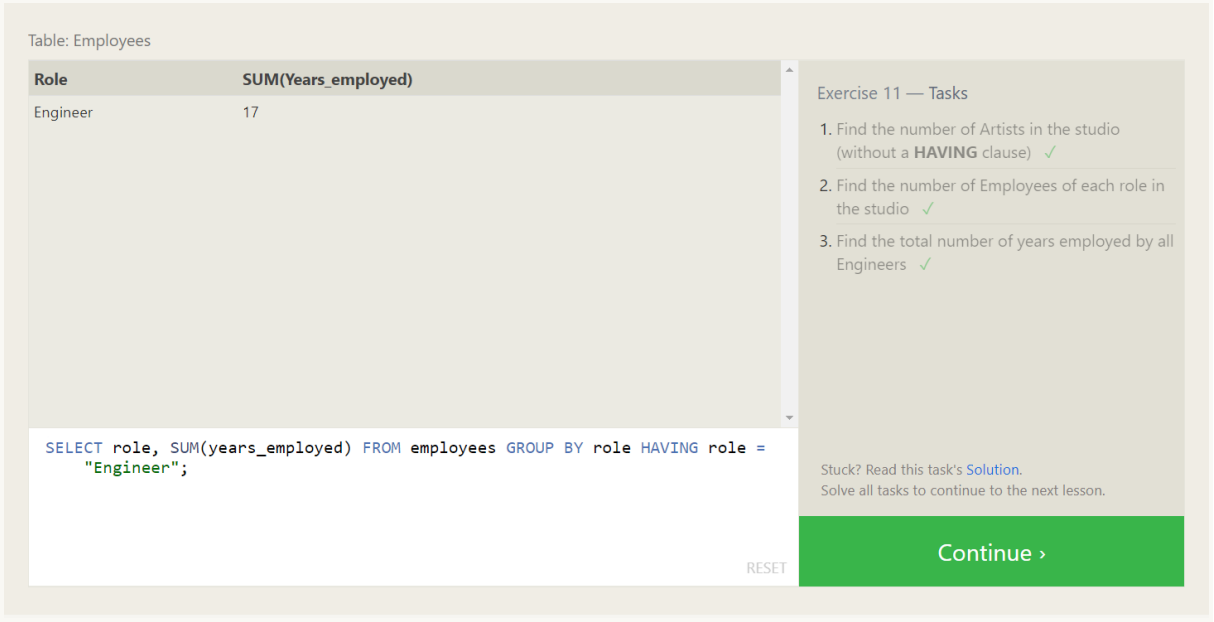
Continue ›

SQL Lesson 11: Queries with aggregates

1)Find the number of Artists in the studio (without a **HAVING** clause) :SELECT role, COUNT(*) as Number_of_artists FROM employees WHERE role = "Artist";

2)Find the number of Employees of each role in the studio:SELECT role, COUNT(*)FROM employees GROUP BY role;

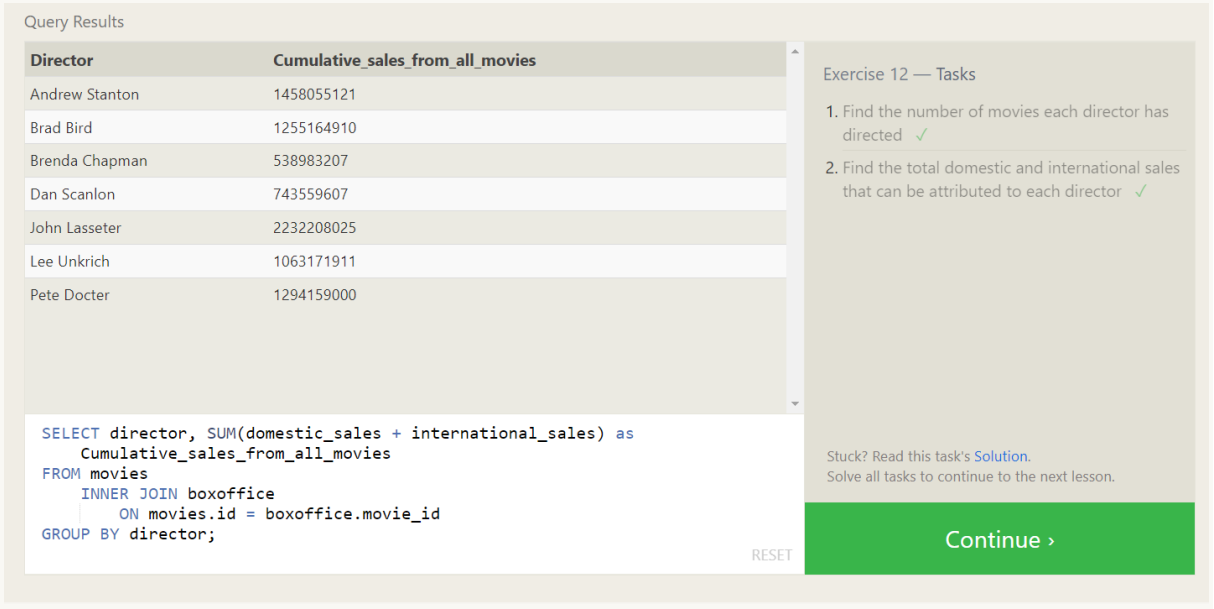
3)Find the total number of years employed by all Engineers :SELECT role, SUM(years_employed) FROM employees GROUP BY role HAVING role = "Engineer";



SQL Lesson 12: Order of execution of a Query

1)Find the number of movies each director has directed : SELECT director, COUNT(id) as Num_movies_directed FROM movies GROUP BY director;

2)Find the total domestic and international sales that can be attributed to each director: 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;



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 ✓

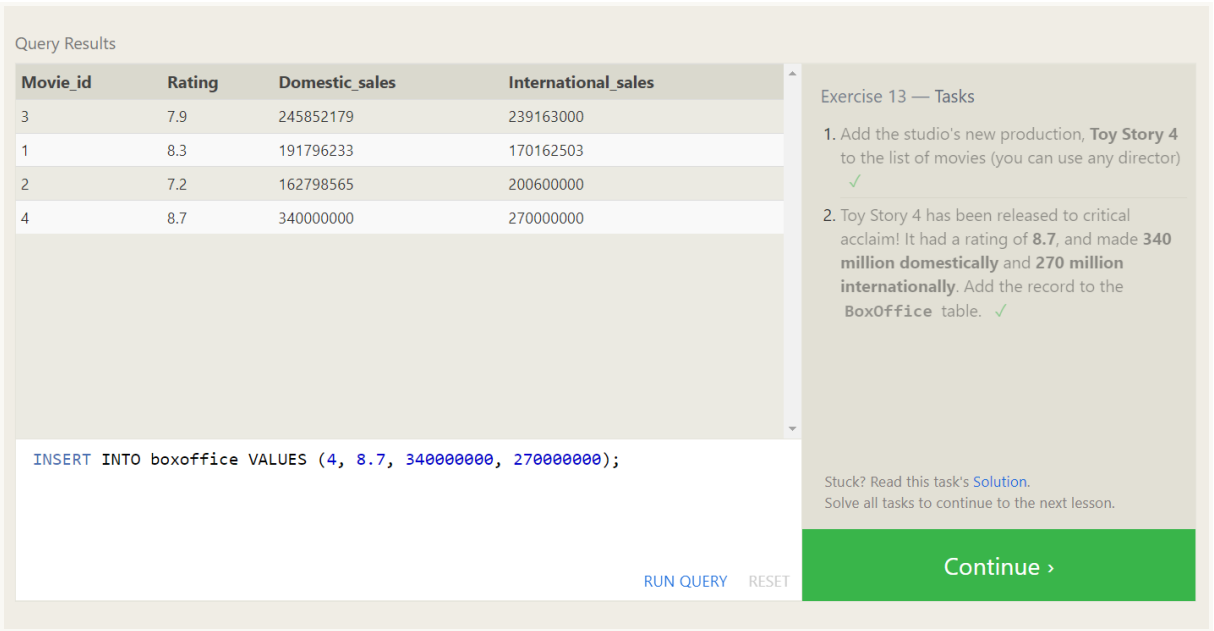
Stuck? Read this task's Solution.
Solve all tasks to continue to the next lesson.

Continue ›

SQL Lesson 13: Inserting rows

1) Add the studio's new production, **Toy Story 4** to the list of movies(you can use any director) : INSERT INTO movies VALUES (4, "Toy Story 4", "El Directore", 2015, 90);

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: INSERT INTO boxoffice VALUES (4, 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
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.

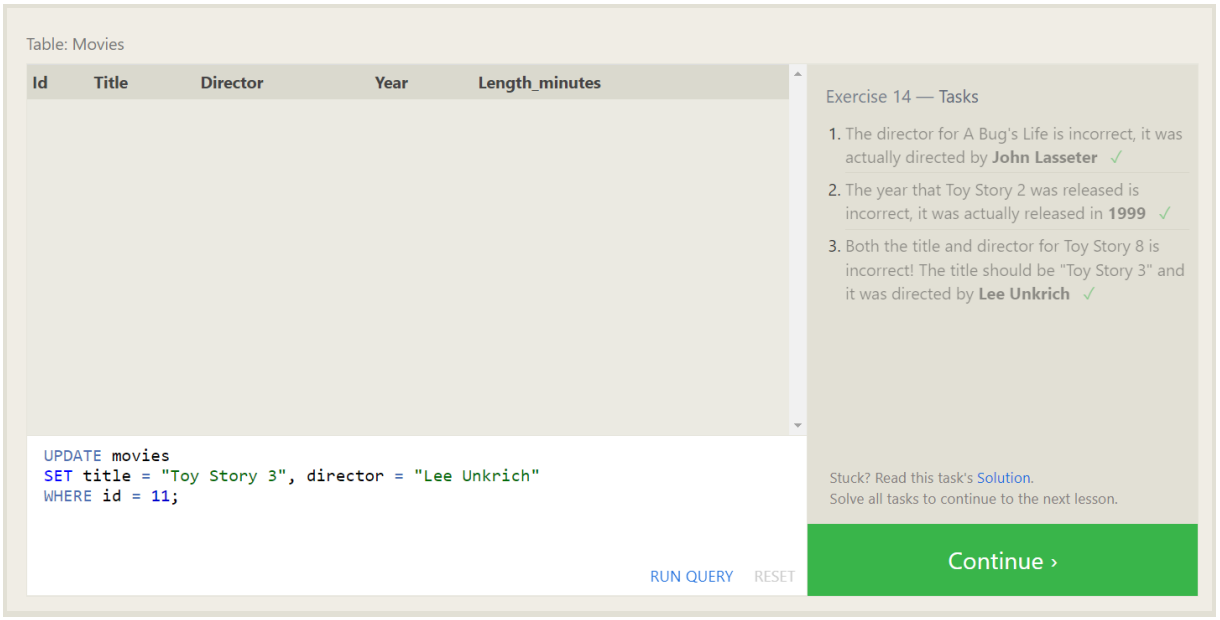
Continue ›

SQL Lesson 14: Updating rows

1) The director for A Bug's Life is incorrect, it was actually directed by **John Lasseter: UPDATE movies SET director = "John Lasseter" WHERE id = 2;**

2)The year that Toy Story 2 was released is incorrect, it was actually released in 1999: UPDATE movies SET year = 1999 WHERE id = 3;

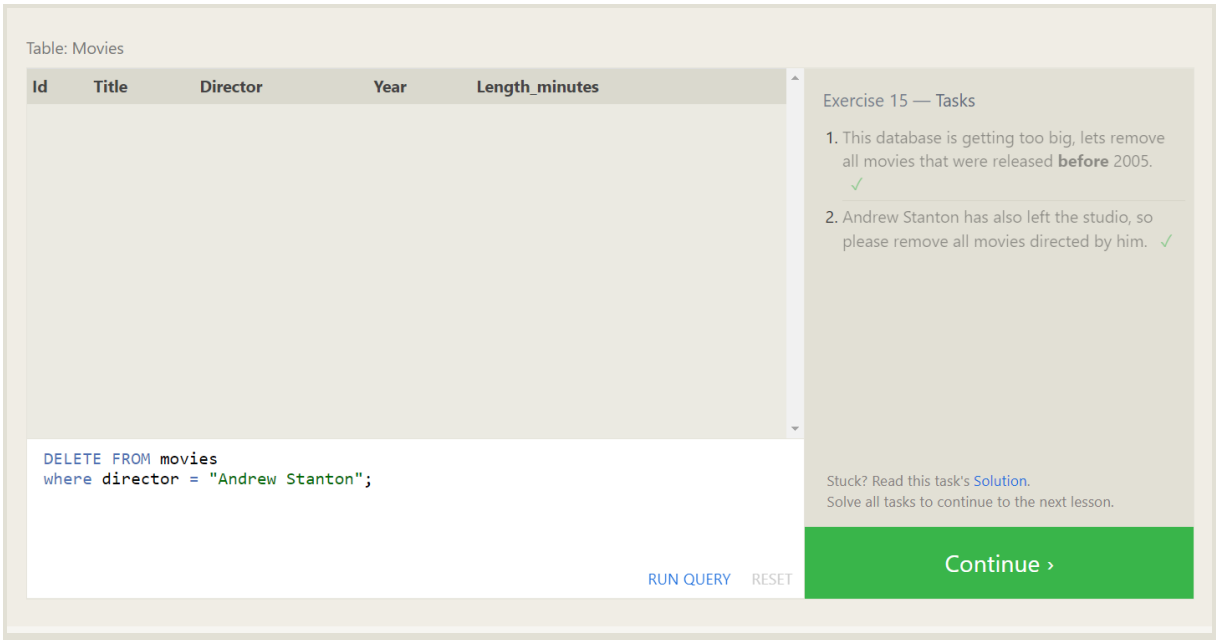
3)Both the title and director for Toy Story 8 is incorrect! The title should be "Toy Story 3" and it was directed by Lee Unkrich: UPDATE movies SET title = "Toy Story 3", director = "Lee Unkrich" WHERE id = 11;



SQL Lesson 15: Deleting rows

1)This database is getting too big, lets remove all movies that were released **before** 2005 : DELETE FROM movies where year < 2005;

2) Andrew Stanton has also left the studio, so please remove all movies directed by him. : DELETE FROM movies where director = "Andrew Stanton";



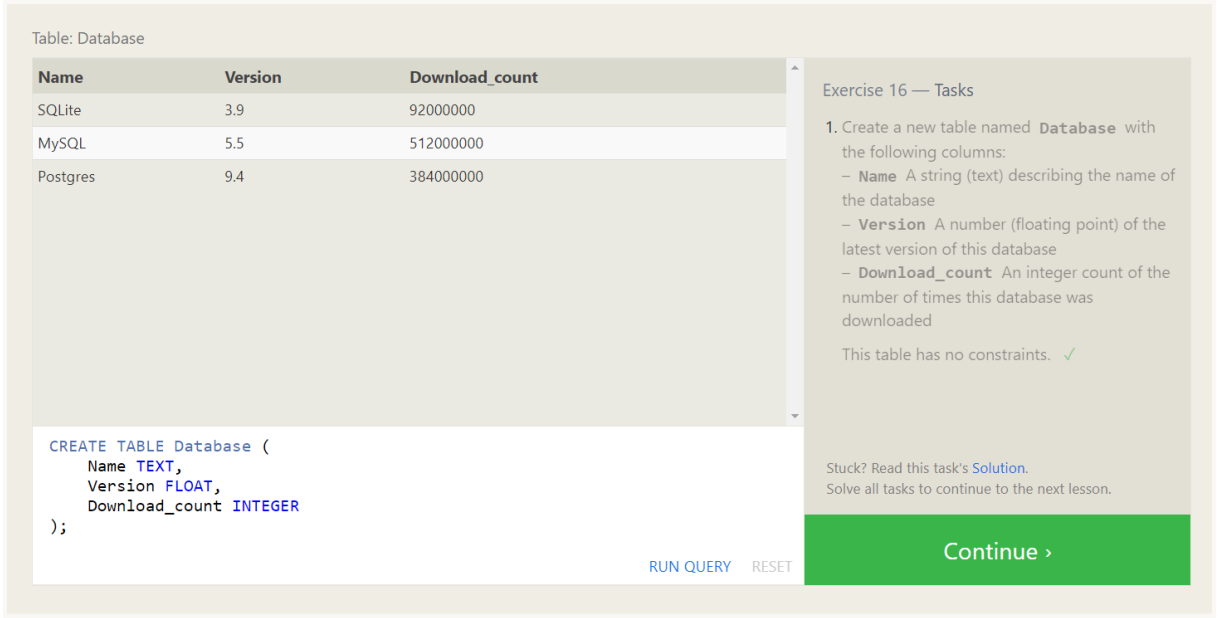
SQL Lesson 16: Creating tables

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

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



SQL Lesson 17: Altering tables

1) Add a column named **Aspect_ratio** with a **FLOAT** data type to store the aspect-ratio each movie was released in. : ALTER TABLE Movies ADD COLUMN Aspect_ratio FLOAT DEFAULT 2.39;

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 : ALTER TABLE Movies ADD COLUMN Language TEXT DEFAULT "English";**

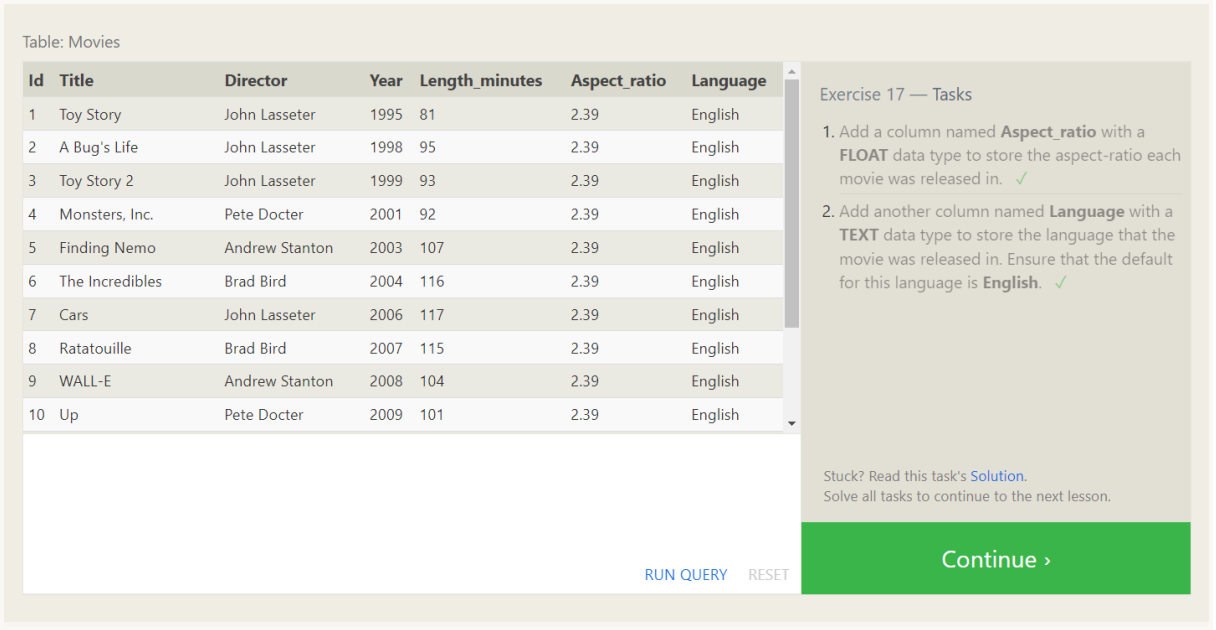


Table: Movies

Id	Title	Director	Year	Length_minutes	Aspect_ratio	Language
1	Toy Story	John Lasseter	1995	81	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 ›

SQL Lesson 18: Dropping tables

1)We've sadly reached the end of our lessons,lets clean up by removing the **Movies** table:drop table movies;

2)And drop the **BoxOffice** table as well:drop table boxoffice ;

Query Results

Id	Title	Director	Year	Length_minutes

```
DROP TABLE BoxOffice;
```

RUN QUERY RESET

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.

Continue ›