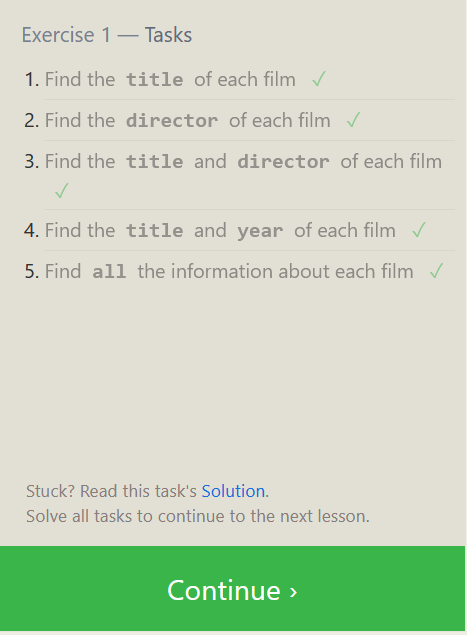
# SQL ASSIGNMENT

## EXERCISE 1 TASKS



1. Find the title of each film

*SELECT TITLE FROM movies;*

1. Find the director of each film

*SELECT DIRECTOR FROM movies;*

1. Find the title and director of each film

*SELECT TITLE,DIRECTOR FROM movies;*

1. Find the title and year of each film

*SELECT TITLE,YEAR FROM movies;*

1. Find all the information about each film

*SELECT \* FROM movies;*

*A screenshot of a phone

Description automatically generated*

## EXERCISE 2 TASKS

1. Find the movie with a row id of 6

*SELECT \* FROM movies where id=6;*

1. Find the movies released in the years between 2000 and 2010

*SELECT \* FROM movies where year>1999 and year<2010;*

1. Find the movies **not** released in the years between 2000 and 2010

*SELECT \* FROM movies WHERE year < 2000 OR year > 2010;*

1. Find the first 5 Pixar movies and their release year

*SELECT \* FROM movies,year where id<=5;*

## *A screenshot of a phone Description automatically generated*

## EXERCISE 3 TASKS

1. Find all the Toy Story movies

*SELECT \* FROM movies where title LIKE "%Toy%";*

1. Find all the movies directed by John Lasseter

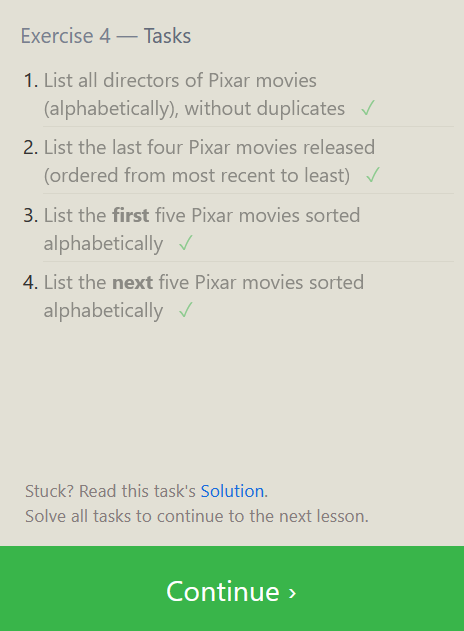
*SELECT \* FROM movies where director LIKE "%John%";*

1. Find all the movies (and director) not directed by John Lasseter

*SELECT \* FROM movies director where director NOT LIKE "%John%";*

1. Find all the WALL-\* movies

*SELECT \* FROM movies where title LIKE "wall%";*



EXERCISE 4 TASKS

1. List all directors of Pixar movies (alphabetically), without duplicates

*SELECT DISTINCT director FROM Movies ORDER BY director asc;*

1. List the last four Pixar movies released (ordered from most recent to least)

*SELECT \* FROM MOVIES ORDER BY YEAR DESC LIMIT 4;*

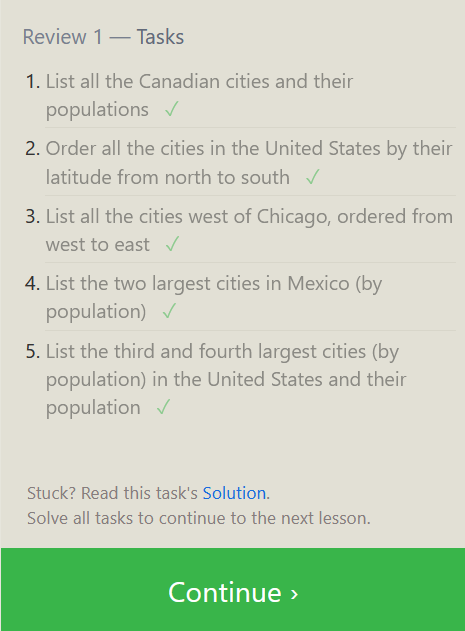
1. List the **first** five Pixar movies sorted alphabetically

*SELECT \* FROM MOVIES ORDER BY title ASC LIMIT 5;*

1. List the **next** five Pixar movies sorted alphabetically

SELECT \* FROM MOVIES ORDER BY title ASC LIMIT 5

OFFSET 5;

**

REVIEW 1 TASKS

1. List all the Canadian cities and their populations

*SELECT City, POPULATION FROM north\_american\_cities*

*WHERE Country='Canada';*

1. Order all the cities in the United States by their latitude from north to south

*SELECT city FROM north\_american\_cities where*

*country='United States' order by latitude desc;*

1. List all the cities west of Chicago, ordered from west to east

*SELECT city FROM north\_american\_cities*

*WHERE longitude < -87.6298;*

1. List the two largest cities in Mexico (by population)

*SELECT \* FROM north\_american\_cities WHERE Country='Mexico'*

*order by population desc limit 2;*

1. List the third and fourth largest cities (by population) in the United States and their population

*SELECT \* FROM north\_american\_cities WHERE Country='United States'*

*order by population desc limit 2 offset 2;*

A screenshot of a phone

Description automatically generated

EXERCISE 6 TASKS

1. Find the domestic and international sales for each movie

*SELECT movies.title, boxoffice.domestic\_sales,*

*boxoffice.international\_sales FROM movies JOIN boxoffice*

*ON movies.id = boxoffice.movie\_id;*

1. Show the sales numbers for each movie that did better internationally

rather than domestically

*SELECT movies.title, boxoffice.domestic\_sales,*

*boxoffice.international\_sales FROM movies JOIN*

*boxoffice ON movies.id = boxoffice.movie\_id where domestic\_sales<international\_sales;*

1. List all the movies by their ratings in descending order

*SELECT movies.title, boxoffice.domestic\_sales,*

*boxoffice.international\_sales FROM movies JOIN*

*boxoffice ON movies.id = boxoffice.movie\_id*

*order by rating desc*

A screenshot of a phone

Description automatically generated

EXERCISE 7 TASKS

1. Find the list of all buildings that have employees

*select distinct building employee from employees;*

1. Find the list of all buildings and their capacity ✓

*select distinct building\_name, capacity from buildings;*

1. List all buildings and the distinct employee roles in each

building (including empty buildings)

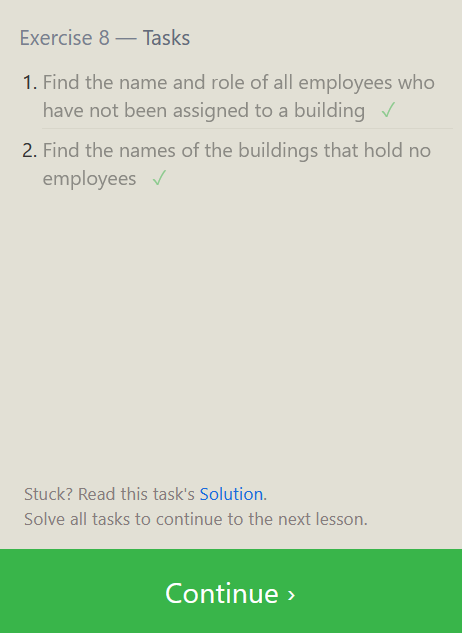
*SELECT b.building\_name, e.role, 'No Employees' AS role*

*FROM buildings b LEFT JOIN*

*(SELECT DISTINCT building AS building\_name,*

*role FROM employees) e ON b.building\_name*

*= e.building\_name ORDER BY b.building\_name, e.role;*

**

EXERCISE 8 TASKS

1. Find the name and role of all employees who have not been

assigned to a building

*SELECT name, role FROM employees WHERE building*

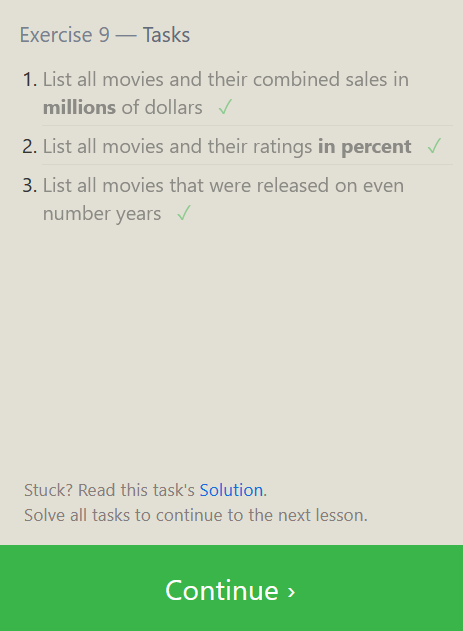
*IS NULL OR building = '';*

1. Find the names of the buildings that hold no employees

*SELECT b.building\_name FROM buildings b*

*LEFT JOIN employees e ON b.building\_name*

*= e.building WHERE e.building IS NULL;*

**

EXERCISE 9 TASKS

1. List all movies and their combined sales in **millions** of dollars

*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;*

1. List all movies and their ratings **in percent**

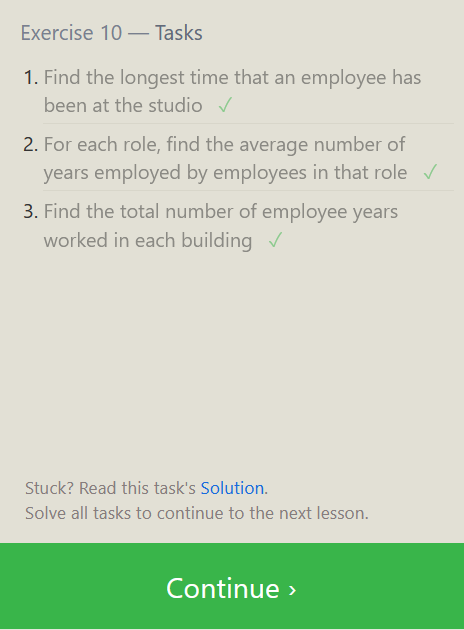
*SELECT m.title, (rating+'%') AS rating\_percent*

*FROM movies m LEFT JOIN boxOffice b*

*ON m.id = b.movie\_id;*

1. List all movies that were released on even number years

*select m.title from movies m where (year%2==0);*



EXERCISE 10 TASKS

1. Find the longest time that an employee has been at the studio.

*SELECT max(years\_employed) FROM employees ;*

1. For each role, find the average number of years employed by

employees in that role

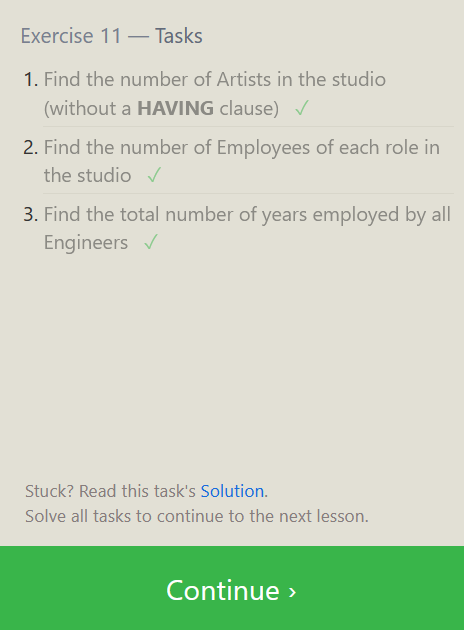
*SELECT role, AVG(years\_employed) FROM employees*

*GROUP BY role;*

1. Find the total number of employee years worked in each building

*select role, building, sum(years\_employed) from employees*

*group by building;*

**

EXERCISE 11 — TASKS

1. Find the number of Artists in the studio (without a **HAVING** clause)

*SELECT COUNT(\*) AS number\_of\_artists*

*FROM employees WHERE role = 'Artist';*

1. Find the number of Employees of each role in the studio

*SELECT role, COUNT(\*) AS number\_of\_employees*

*FROM employees GROUP BY role;*

1. Find the total number of years employed by all Engineers

*SELECT SUM(years\_employed) AS total\_years\_employed*

*FROM employees WHERE role = 'Engineer';*

A screenshot of a computer

Description automatically generatedEXERCISE 12 — TASKS

1. Find the number of movies each director has directed

*SELECT director, COUNT(\*) AS number\_of\_movies*

*FROM movies GROUP BY director;*

1. Find the total domestic and international sales that can be

attributed to each director

SELECT director, SUM(domestic\_sales + international\_sales)

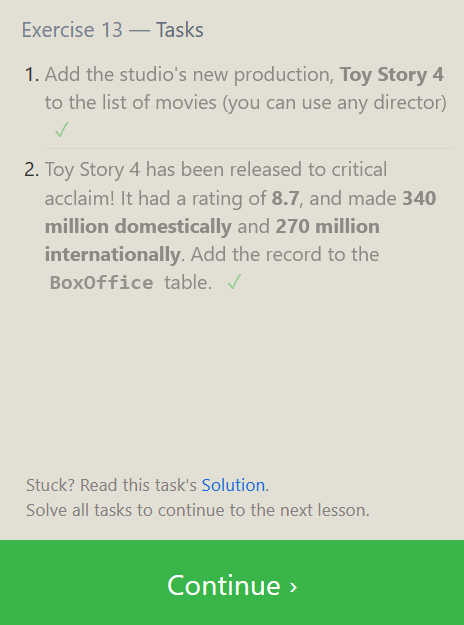
as Total\_Sales

FROM movies

INNER JOIN boxoffice

ON movies.id = boxoffice.movie\_id

GROUP BY director;

**

EXERCISE 13 — TASKS

1. Add the studio's new production, **Toy Story 4** to the list

of movies (you can use any director)

*INSERT INTO MOVIES('id', 'title', 'director', 'year', 'length\_minutes')*

*values(4, 'Toy Story 4', 'John Lasseter', 2000, 98);*

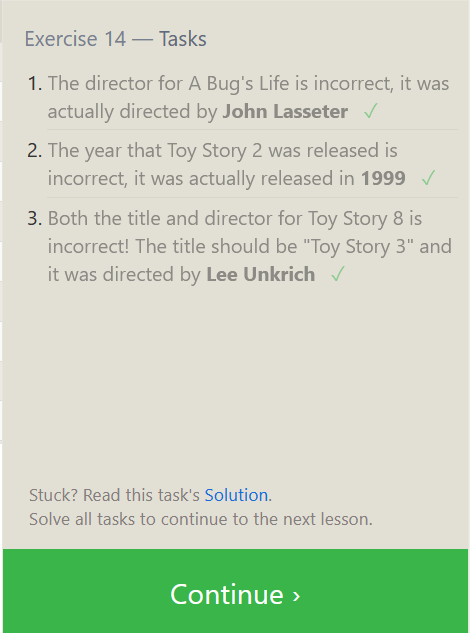
2. Toy Story 4 has been released to critical acclaim! It had a rating

of 8.7, and made 340 million domestically and 270 million

internationally. Add the record to the BoxOffice table.

*INSERT INTO BoxOffice ('movie\_id', 'rating', 'domestic\_sales',*

*'international\_sales') VALUES (4, 8.7, 340, 270);*

**

EXERCISE 14 — TASKS

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;*

1. The year that Toy Story 2 was released is incorrect, it

was actually released in **1999**

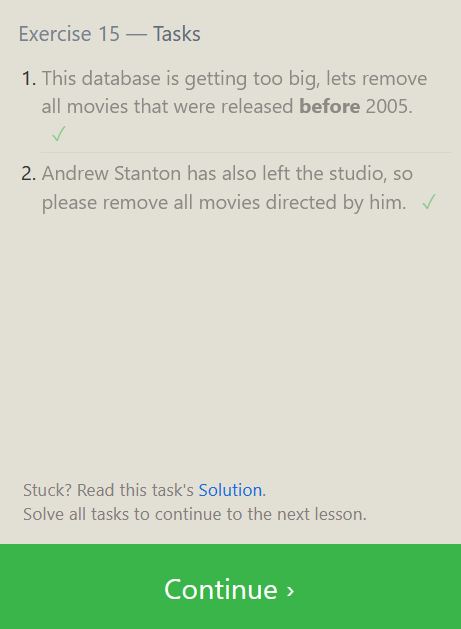
*update movies set year=1999 where id=3;*

1. 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;*

**

EXERCISE 15 — TASKS

1. This database is getting too big, lets remove all movies

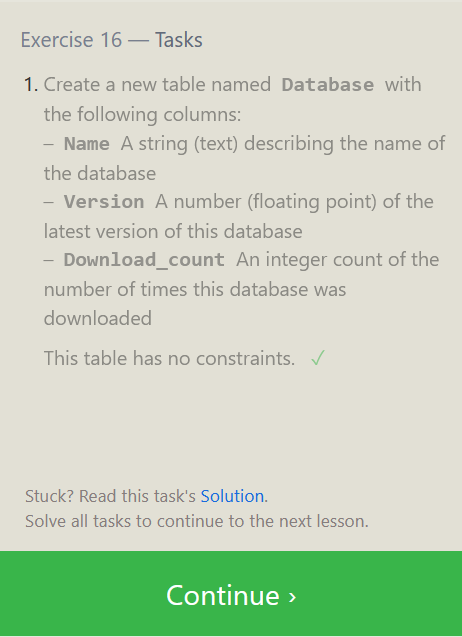
that were released **before** 2005

*DELETE Movies where year<2005;*

1. Andrew Stanton has also left the studio, so please

remove all movies directed by him.

*DELETE FROM MOVIES WHERE Director='Andrew Stanton';*

**

EXERCISE 16 — TASKS

1. Create a new table named Database with the following columns:

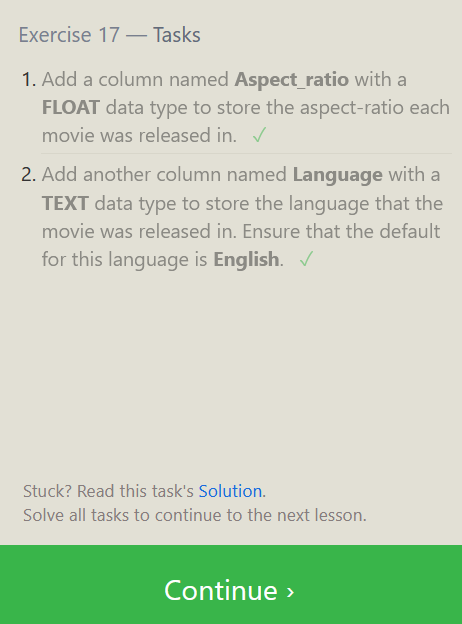
– Name A string (text) describing the name of the database  
– Version A number (floating point) of the latest version of this database  
– Download\_count An integer count of the number of

times this database was downloaded

This table has no constraints.

*create table database(name varchar(255), version float(10, 2),*

*download\_count integer);*



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.

*ALTER TABLE Movies ADD Aspect\_ratio Float;*

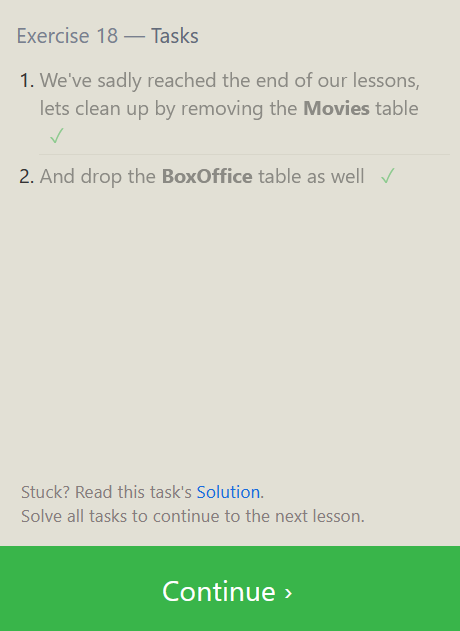
1. Add another column named **Language** with a **TEXT** data type to

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';*

**EXERCISE 18 — TASKS

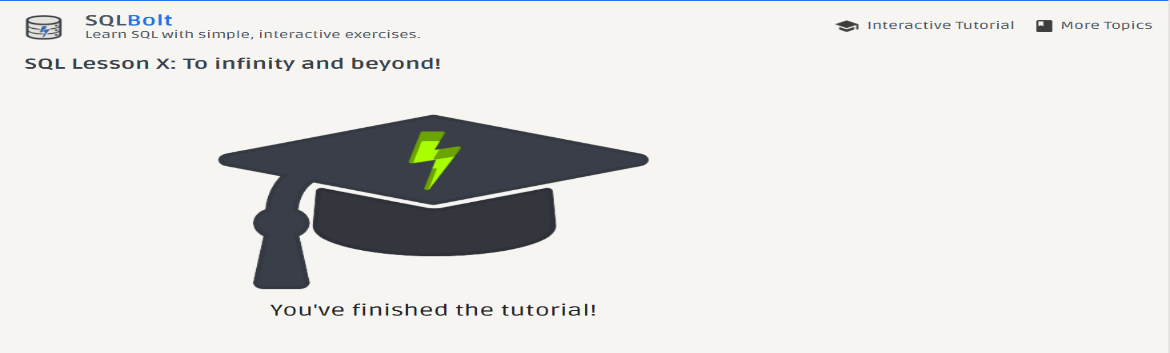
1. We've sadly reached the end of our lessons, lets clean up by removing

the **Movies** table

*DROP TABLE Movies;*

1. And drop the **BoxOffice** table as well

*DROP TABLE BoxOffice;*

**