**SELECT queries 101**

**Exercise 1 - Tasks**

1. Find the title of each film.

SQL Query: **SELECT** title **FROM** movies**;**A screenshot of a computer

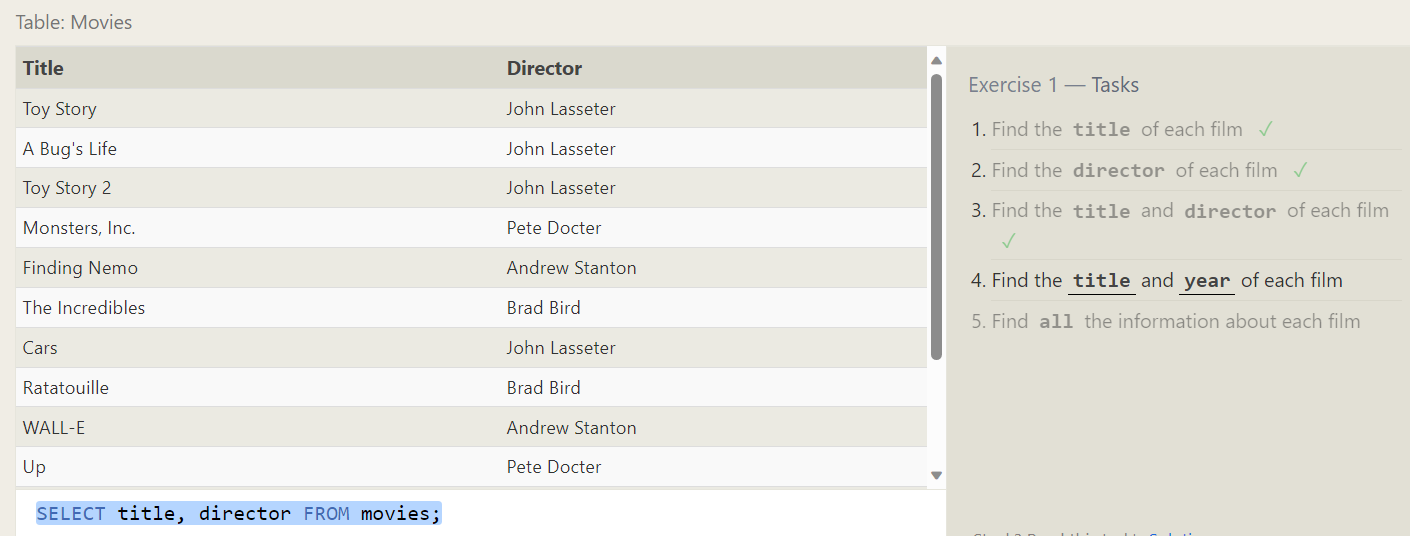
Description automatically generated

1. Find the director of each film.

SQL Query: **SELECT** director **FROM** movies**;**A screenshot of a computer

Description automatically generated

1. Find the title and director of each film.

SQL Query: **SELECT** title**,** director **FROM** movies**;**

1. Find the title and year of each film.

SQL Query: **SELECT** title**,** year **FROM** movies**;**A screenshot of a computer

Description automatically generated

1. Find all the information about each film.

SQL Query: **SELECT** title**,** director **FROM** movies**;**

A screenshot of a computer

Description automatically generated

**Queries with constraints**

**Exercise 2 – Tasks**

1. Find the movie with a row id of 6

**SQL Query: SELECT** **\*** **FROM** movies **WHERE** id **=** 6**;**

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

**SQL Query: SELECT** **\*** **FROM** movies **WHERE** year **BETWEEN** 2000 **AND** 2010**;**

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

**SQL Query: SELECT** **\*** **FROM** movies **WHERE** year **NOT** **BETWEEN** 2000 **AND** 2010**;A screenshot of a computer

Description automatically generated**

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

**SQL Query: SELECT** **\*** **FROM** movies **WHERE** id **BETWEEN** 1 **and** 5**;**A screenshot of a computer

Description automatically generated

**Queries with constraints (Pt. 2)**

**Exercise 3 – Tasks**

1. Find all the Toy Story movies.

**SQL Query: SELECT** **\*** **FROM** movies **WHERE** title **LIKE** "%Toy Story%"**;**A screenshot of a computer

Description automatically generated

1. Find all the movies directed by John Lasseter.

**SQL Query: SELECT** **\*** **FROM** movies **WHERE** director **LIKE** "%John Lasseter%"**;**A screenshot of a computer

Description automatically generated

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

**SQL Query: SELECT** title**,** director **FROM** movies **WHERE** director **NOT** **LIKE** "%John Lasseter%"**;**A screenshot of a computer

Description automatically generated

1. Find all the WALL-\* movies

**SQL Query: SELECT** **\*** **FROM** movies **WHERE** title **LIKE** "%WALL-%"**;**A screenshot of a computer

Description automatically generated

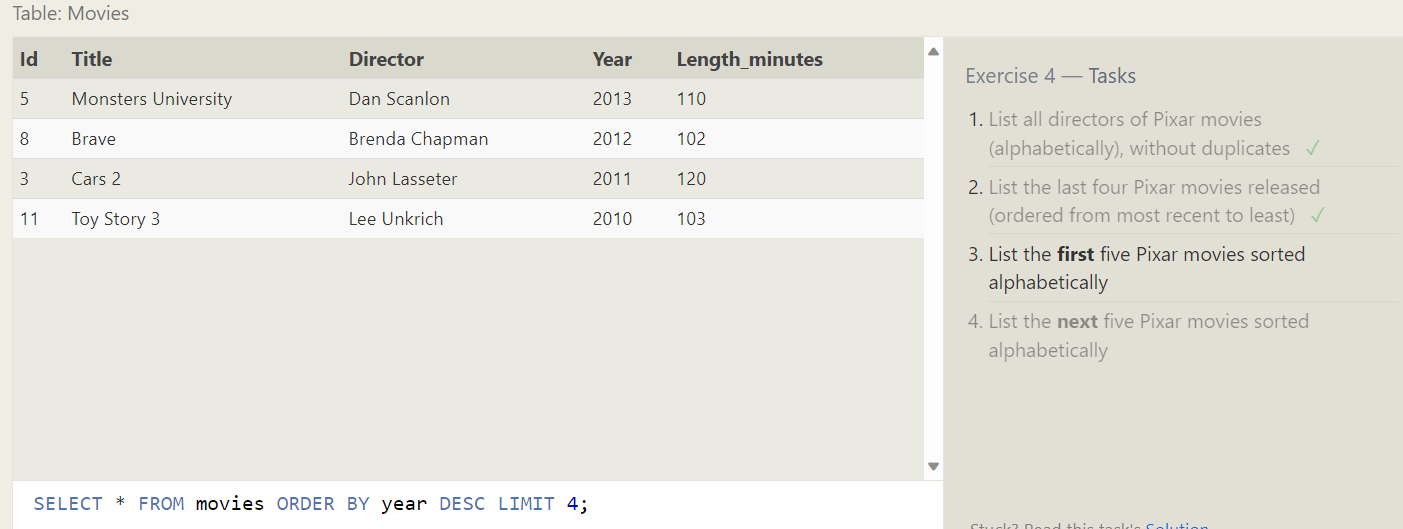
**Filtering and sorting Query results**

**Exercise 4 – Tasks**

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

**SQL Query: SELECT** **DISTINCT** director **FROM** movies **ORDER** **BY** director **ASC;**

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

**SQL Query: SELECT** **\*** **FROM** movies **ORDER** **BY** year **DESC** **LIMIT** 4**;**

1. List the first five Pixar movies sorted alphabetically

**SQL Query: SELECT** **\*** **FROM** movies **ORDER** **BY** title **ASC** **LIMIT** 5**;**

1. List the next five Pixar movies sorted alphabetically

**SQL Query:** **SELECT** **\*** **FROM** movies **ORDER** **BY** title **ASC** **LIMIT** 5 OFFSET 5**;**A screenshot of a computer

Description automatically generated

**Simple SELECT Queries**

**Exercise 5 – Tasks**

1. List all the Canadian cities and their populations

SQL Query: **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

**SQL Query: SELECT** city **FROM** north\_american\_cities **WHERE** country **=**"United States" **ORDER** **BY** latitude **DESC;**A screenshot of a computer

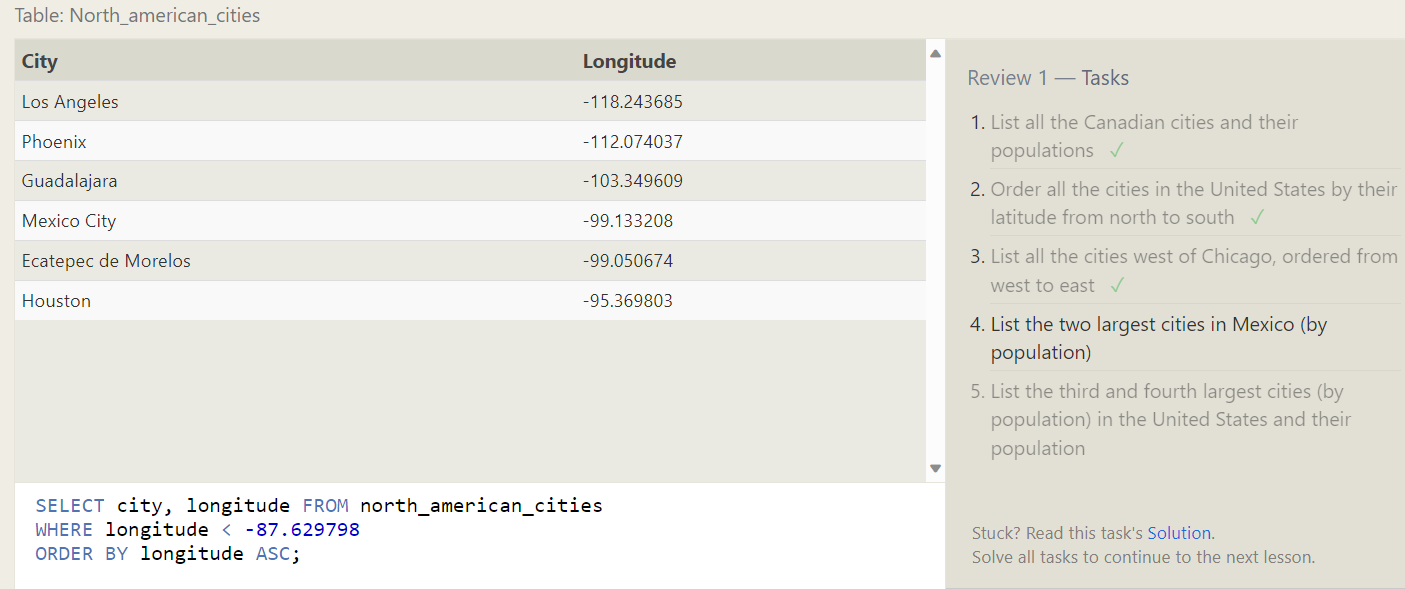
Description automatically generated

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

SQL Query : **SELECT** city**,** longitude **FROM** north\_american\_cities

**WHERE** longitude **<** -87.629798

**ORDER** **BY** longitude **ASC;**



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

SQL Query : **SELECT** **\*** **FROM** north\_american\_cities **WHERE** country **=**"Mexico" **ORDER** **BY** Population **desc** **limit** 2**;**A screenshot of a computer

Description automatically generated

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

SQL Query: **SELECT** **\*** **FROM** north\_american\_cities **WHERE** country **=**"United States" **ORDER** **BY** Population **desc** **limit** 2 OFFSET 2**;**A screenshot of a computer

Description automatically generated

**Simple SELECT Queries**

**Exercise 6 – Tasks**

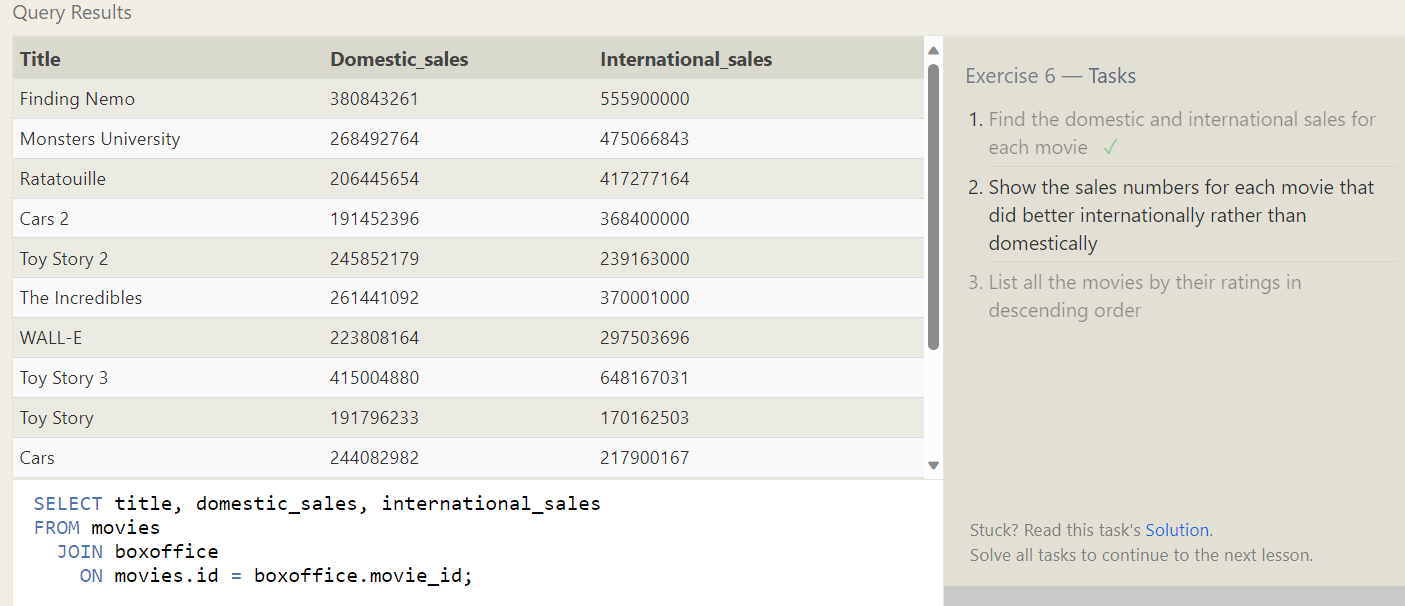
1. Find the domestic and international sales for each movie

SQL Query: **SELECT** title**,** domestic\_sales**,** 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

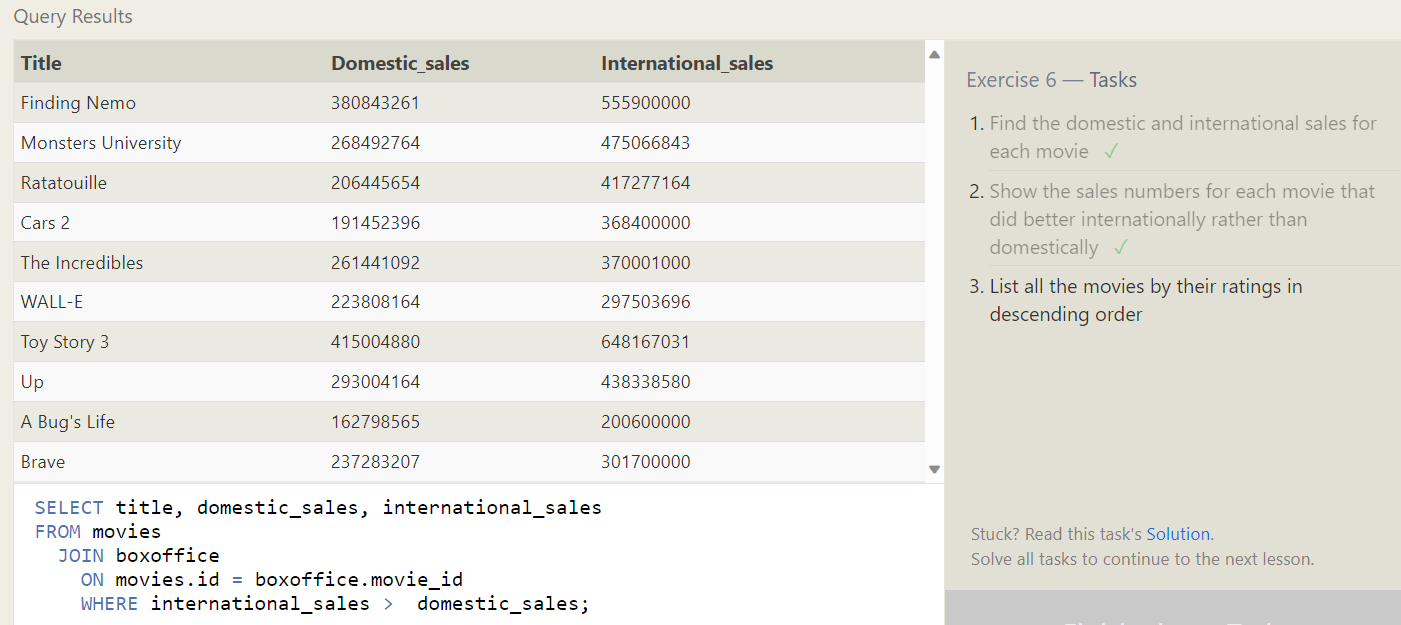
SQL Query: **SELECT** title**,** domestic\_sales**,** international\_sales

**FROM** movies

**JOIN** boxoffice

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

**WHERE** international\_sales **>** domestic\_sales**;**



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

SQL Query: **SELECT** title**,** director**,** year**,** rating

**FROM** movies

**JOIN** boxoffice

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

**ORDER** **BY** rating **desc**

A screenshot of a computer

Description automatically generated

**OUTER JOINs**

**Exercise 7 – Tasks**

1. Find the list of all buildings that have employees

SQL Query: **SELECT** **DISTINCT** Building\_name

**FROM** employees

**LEFT** **JOIN** Buildings

**ON** Buildings**.**Building\_name **=** Employees**.**Building**;**

A screenshot of a computer

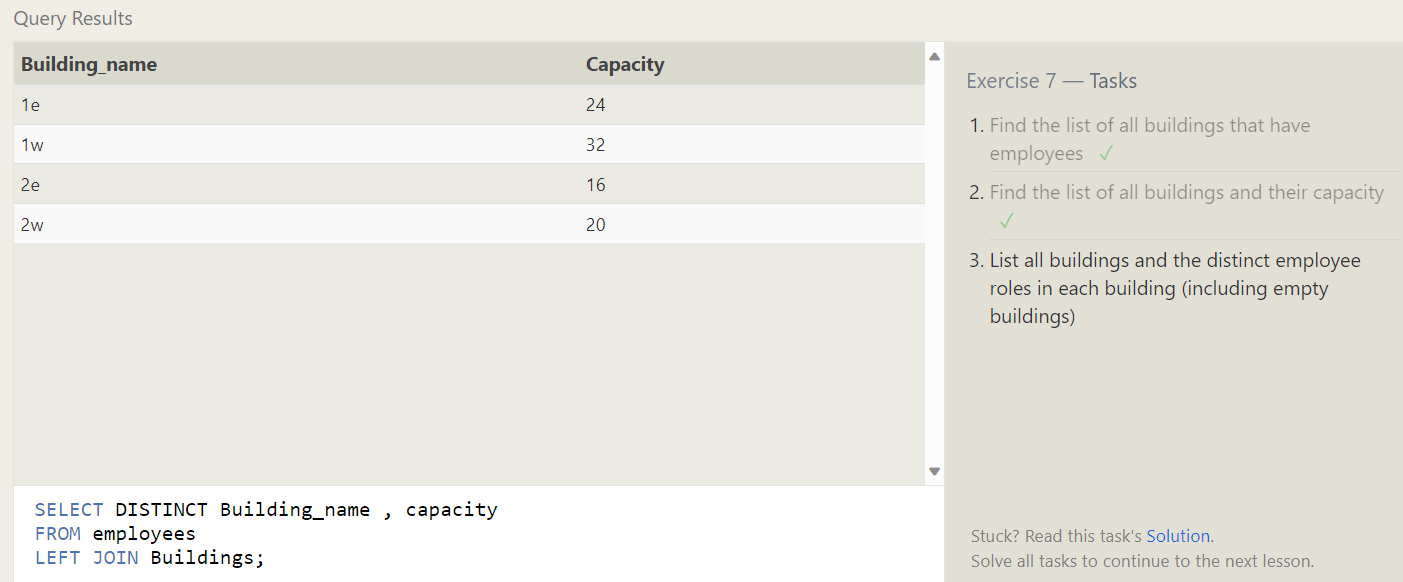
Description automatically generated

1. Find the list of all buildings and their capacity

SQL Query: **SELECT** **DISTINCT** Building\_name **,** capacity

**FROM** employees

**LEFT** **JOIN** Buildings**;**



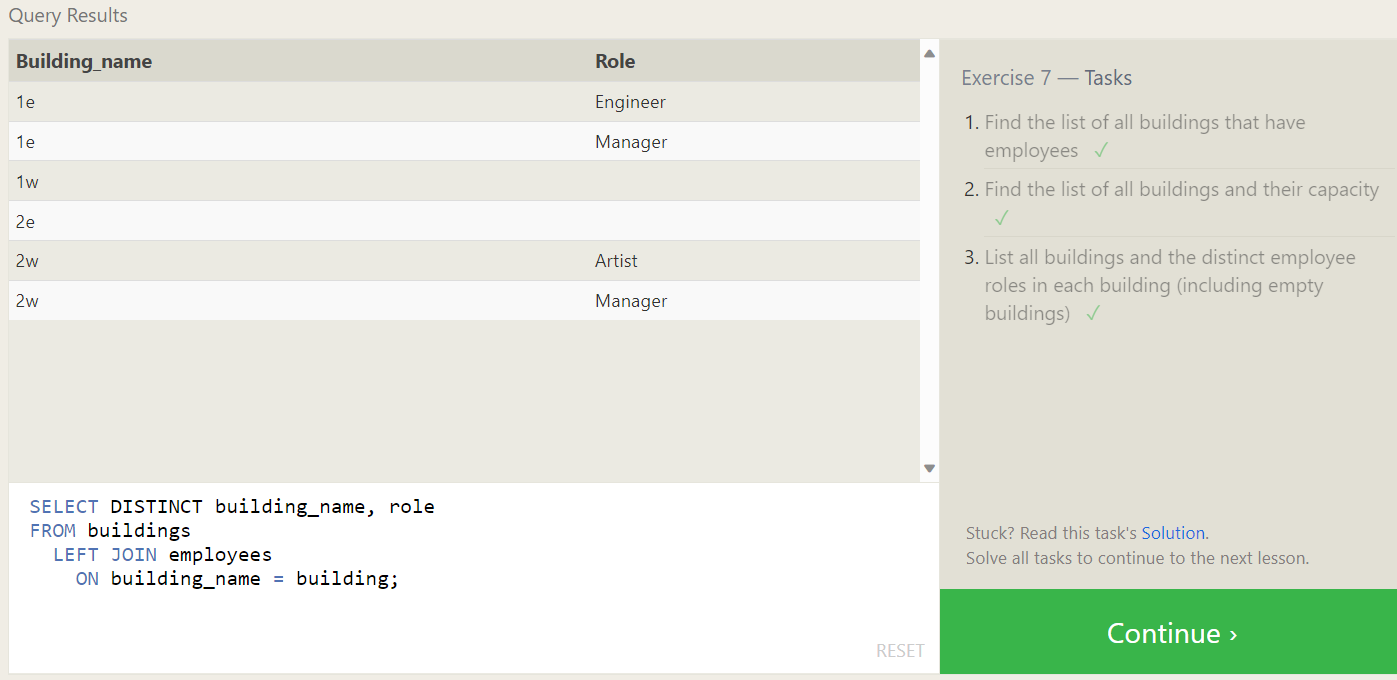
1. List all buildings and the distinct employee roles in each building (including empty buildings)

SQL Query: **SELECT** **DISTINCT** building\_name**,** role

**FROM** buildings

**LEFT** **JOIN** employees

**ON** building\_name **=** building**;**



**A short note on NULLs**

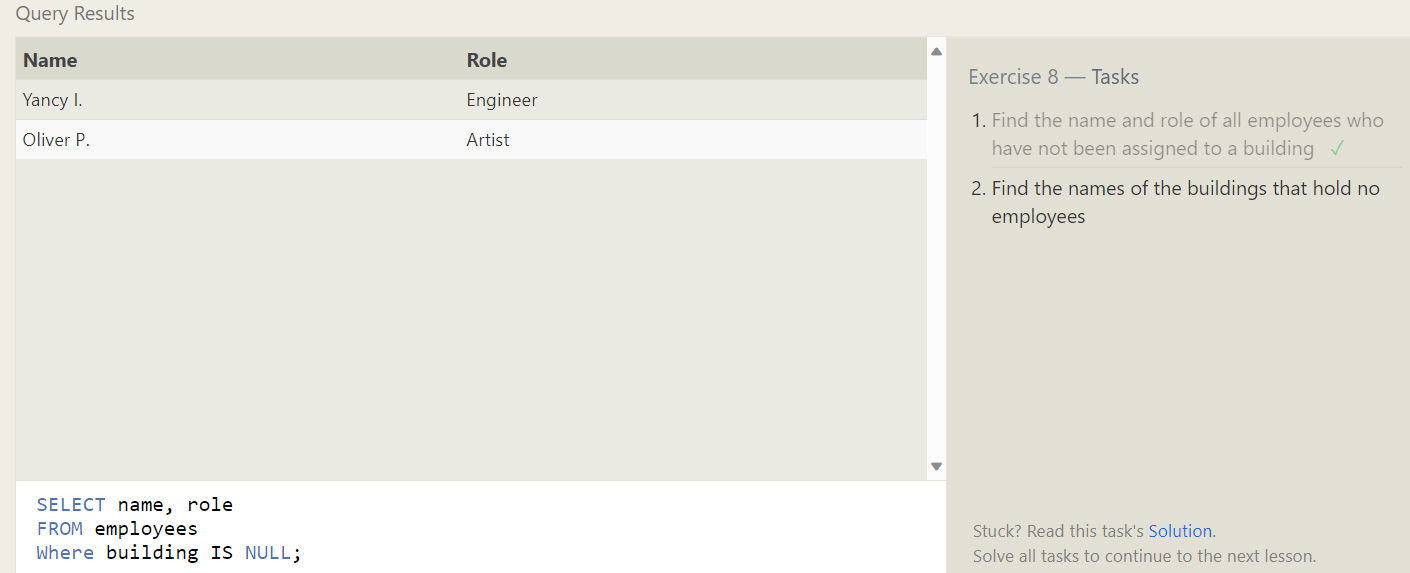
**Exercise 8 – Tasks**

1. Find the name and role of all employees who have not been assigned to a building

SQL Query: **SELECT** name**,** role

**FROM** employees

**Where** building **IS** **NULL;**



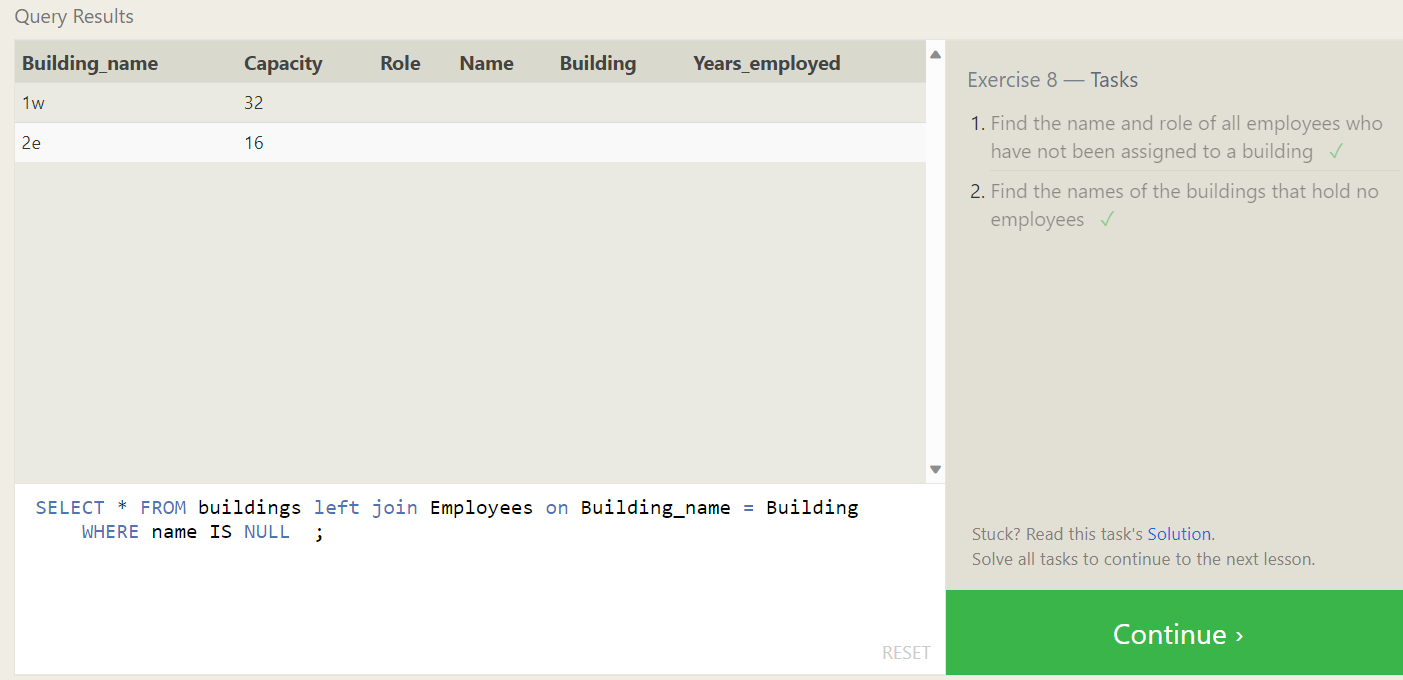
1. Find the names of the buildings that hold no employees

SQL Query: **SELECT** **\*** **FROM** buildings

**left** **join** Employees

**on** Building\_name **=** Building

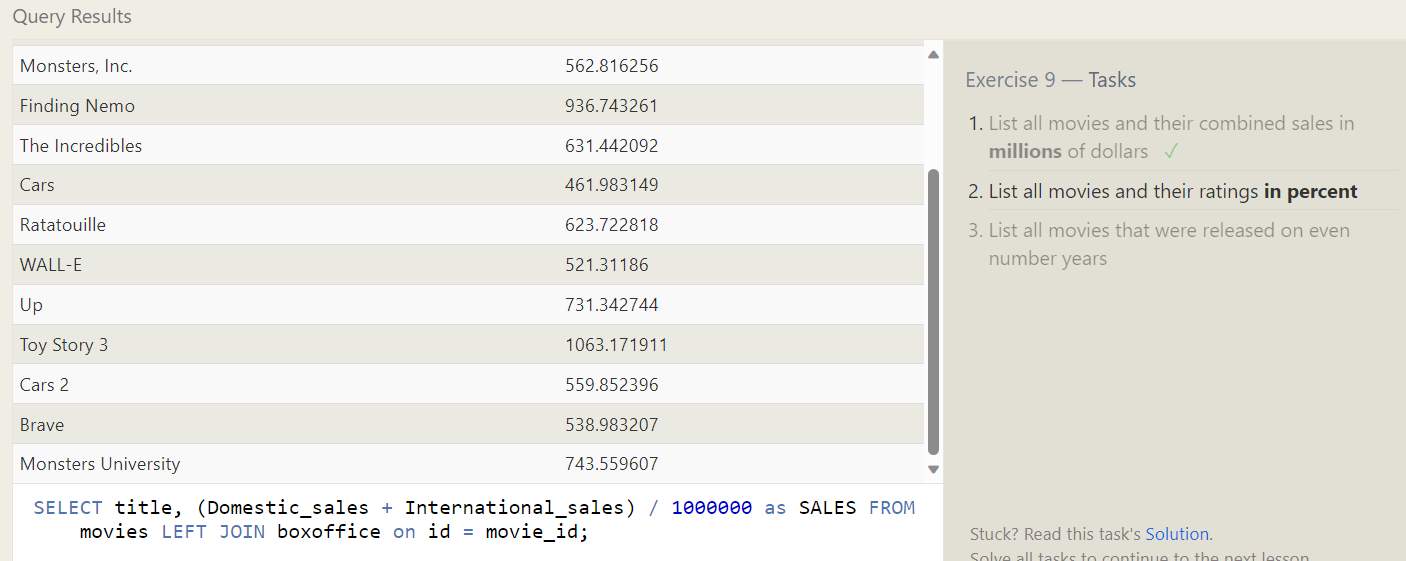
**WHERE** name **IS** **NULL** **;**



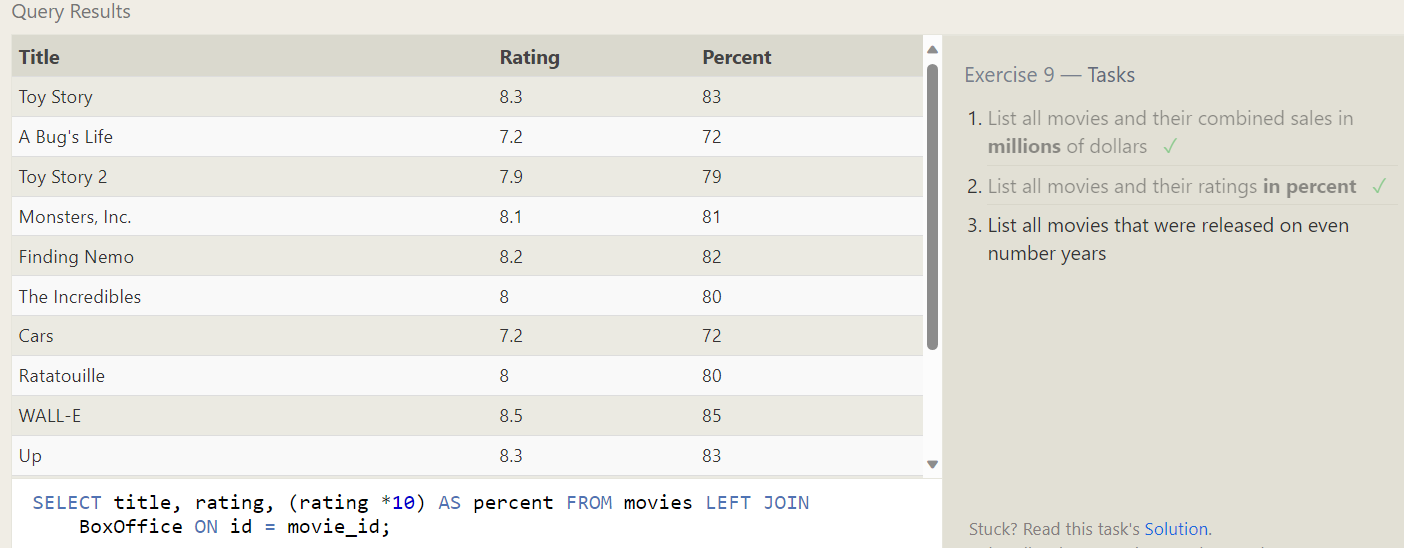
**Queries with expressions**

**Exercise 9 – Tasks**

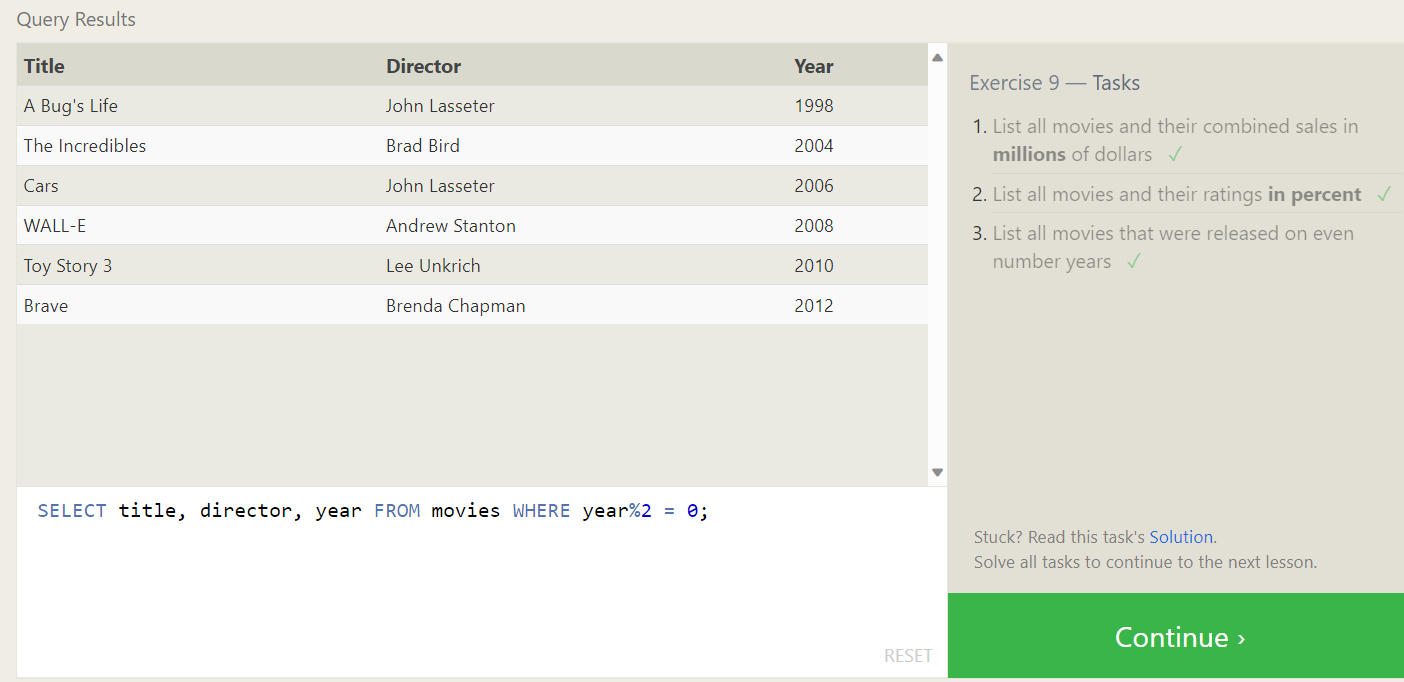
1. List all movies and their combined sales in millions of dollars

SQL Query: **SELECT** title**,** **(**Domestic\_sales **+** International\_sales**)** **/** 1000000 **as** SALES **FROM** movies **LEFT** **JOIN** boxoffice **on** id **=** movie\_id**;**

1. List all movies and their ratings in percent

SQL Query: **SELECT** title**,** rating**,** **(**rating **\***10**)** **AS** **percent** **FROM** movies **LEFT** **JOIN** BoxOffice **ON** id **=** movie\_id**;**

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

SQL Query: **SELECT** title**,** director**,** year **FROM** movies **WHERE** year**%**2 **=** 0**;**

**Queries with aggregates (Pt. 1)**

**Exercise 10 – Tasks**

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

SQL Query: **SELECT** role**,** name**,** **MAX(**Years\_Employed**)** **as** Longest\_Employed **FROM** employees**;**A screenshot of a computer

Description automatically generated

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

SQL Query: **SELECT** role**,** **avg(**years\_employed**)** **as** Average\_Year **FROM** employees **group** **by** role**;**A screenshot of a computer

Description automatically generated

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

SQL Query: **SELECT** building**,** **SUM(**Years\_employed**)** **AS** Total\_Employee **FROM** employees **GROUP** **BY** building**;**A screenshot of a computer

Description automatically generated

**Queries with aggregates (Pt. 2)**

**Exercise 11 – Tasks**

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

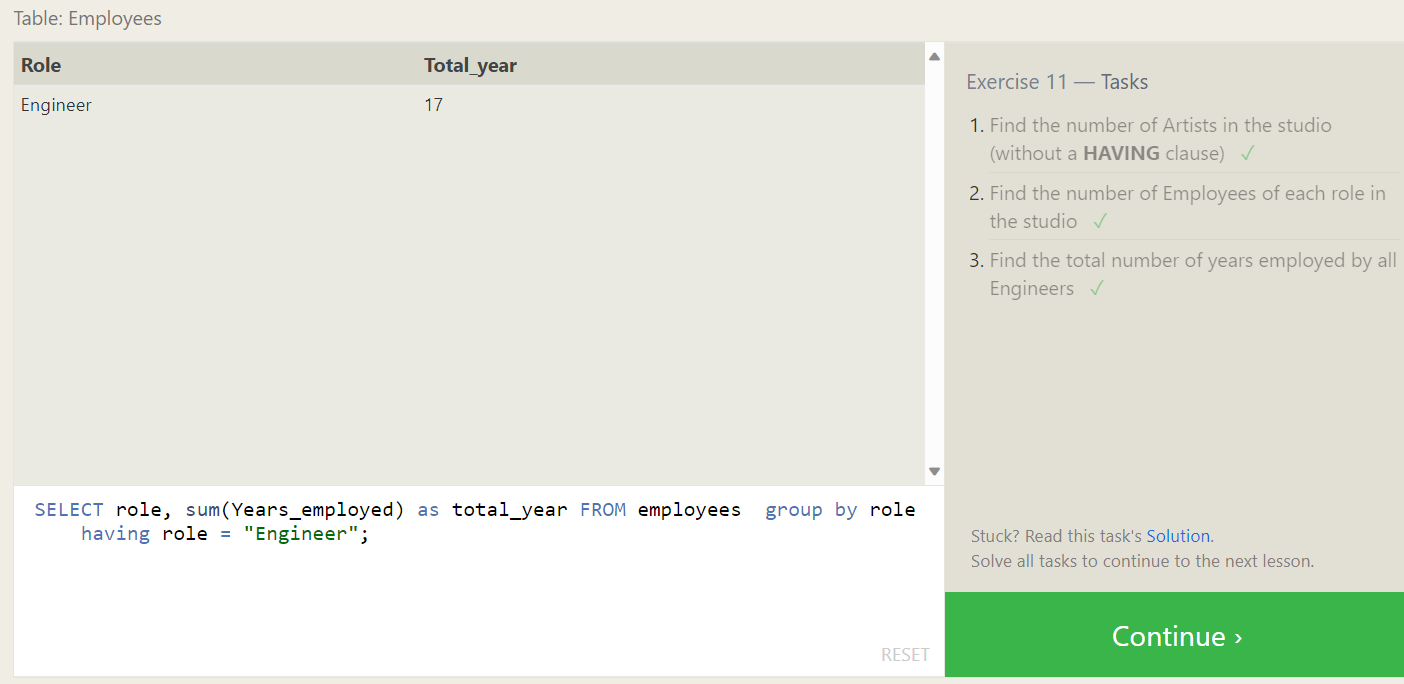
SQL Query: **SELECT** role**,** **count(**role**)** **as** no\_of\_artist **FROM** employees **where** role **=** "Artist"**;**

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

SQL Query: **SELECT** role**,** **count(**name**)** **as** No\_Of\_Employee **FROM** employees **group** **by** role**;**A screenshot of a computer

Description automatically generated

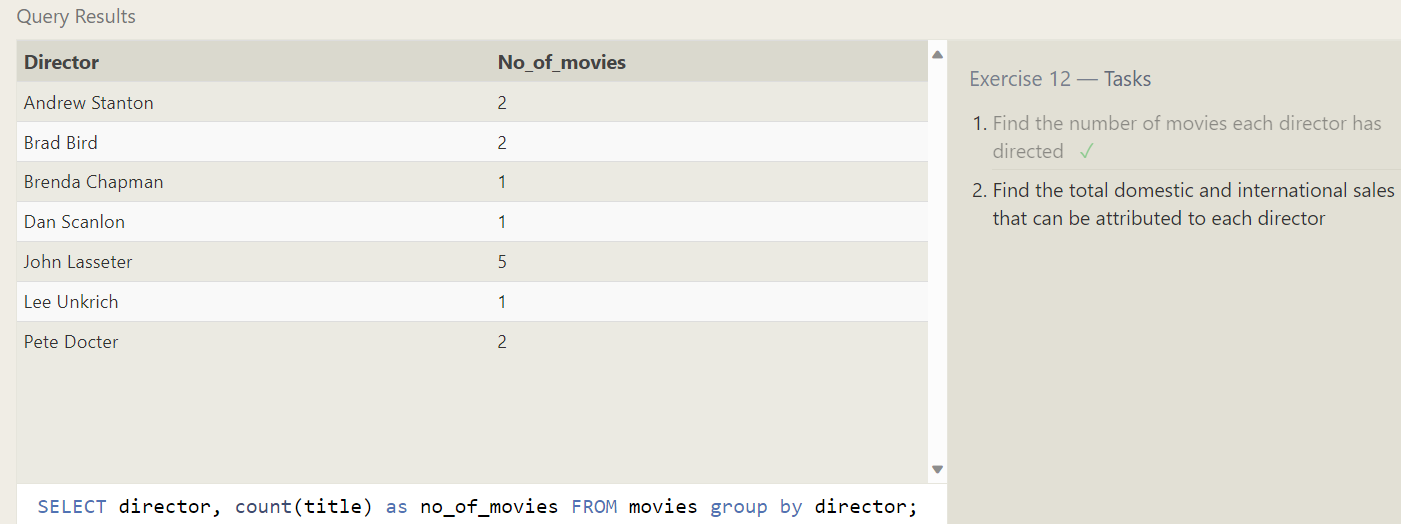
1. Find the total number of years employed by all Engineers

SQL Query: **SELECT** role**,** **sum(**Years\_employed**)** **as** total\_year **FROM** employees **group** **by** role **having** role **=** "Engineer"**;**

**Order of execution of a Query**

**Exercise 12 – Tasks**

1. Find the number of movies each director has directed.

SQL Query: **SELECT** director**,** **count(**title**)** **as** no\_of\_movies **FROM** movies **group** **by** director**;**

1. Find the total domestic and international sales that can be attributed to each director.

SQL Query: **SELECT** director**,**Domestic\_sales**,**International\_sales**,**

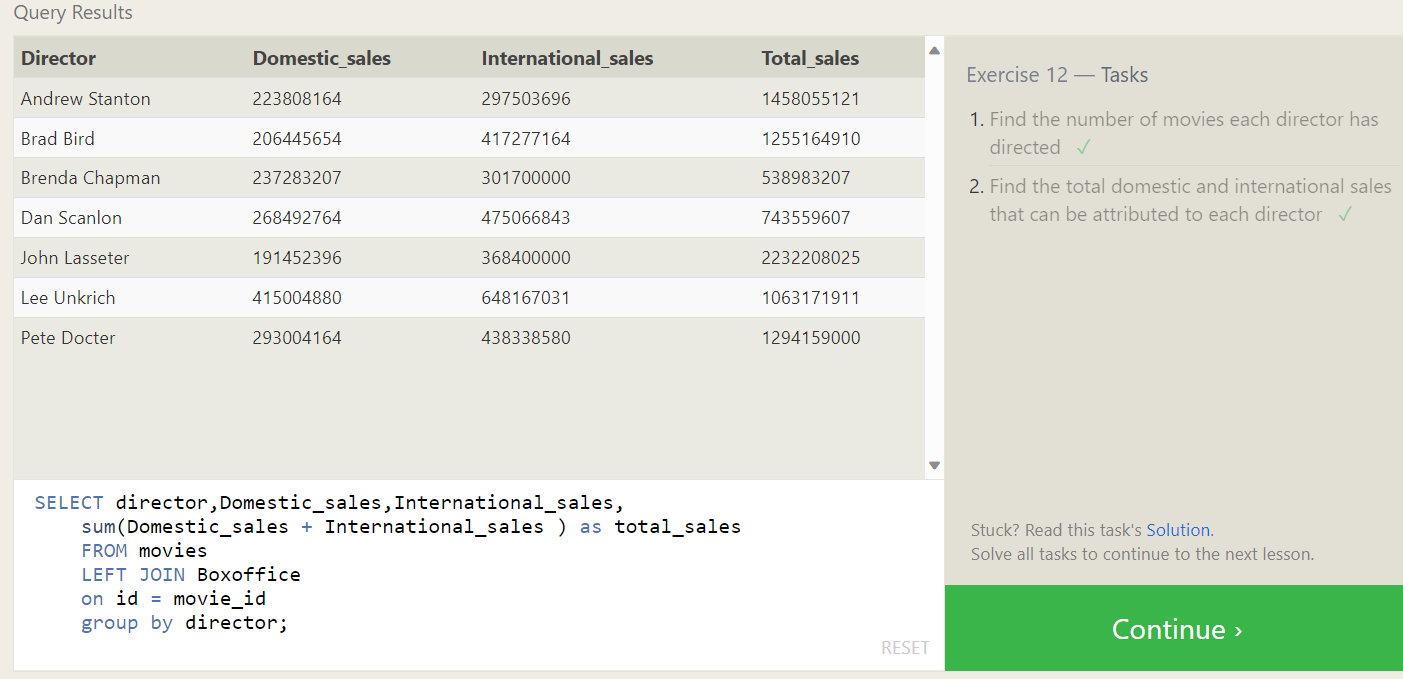
**sum(**Domestic\_sales **+** International\_sales **)** **as** total\_sales

**FROM** movies

**LEFT** **JOIN** Boxoffice

**on** id **=** movie\_id

**group** **by** director**;**



**Inserting rows**

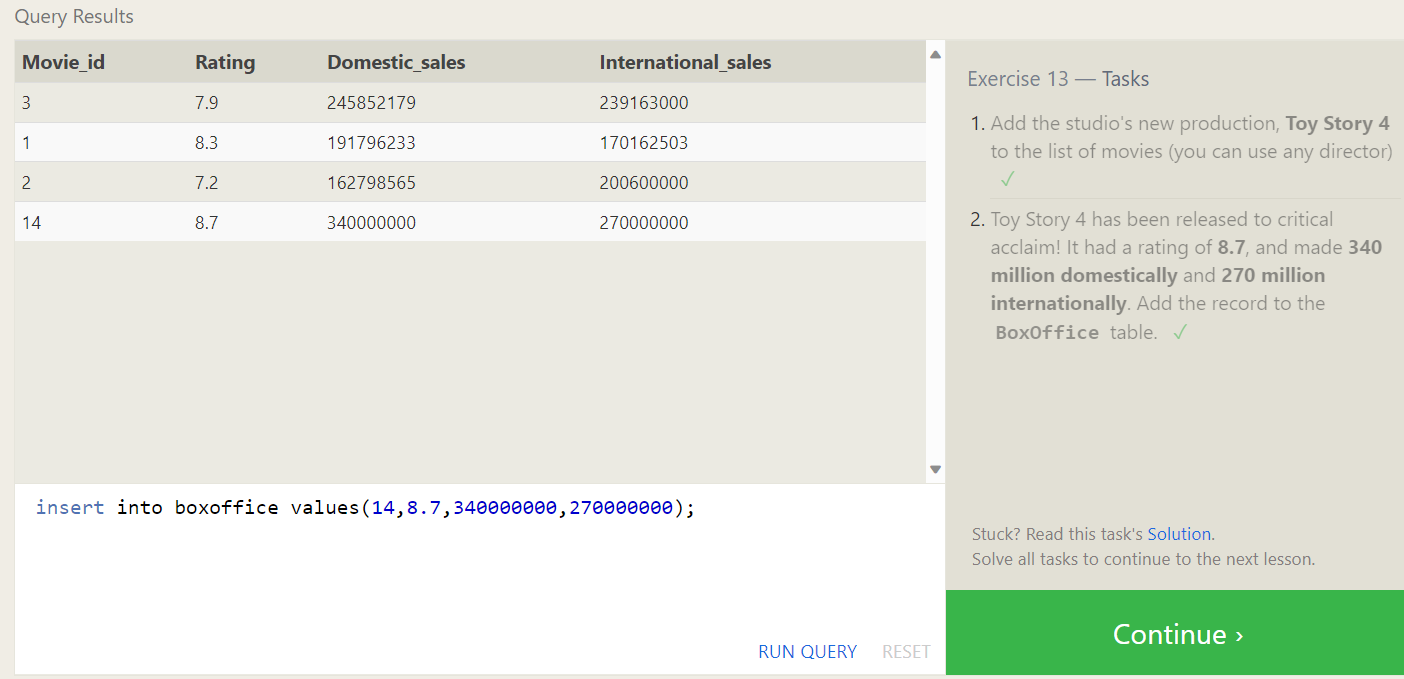
**Exercise 13 – Tasks**

1. Add the studio's new production, Toy Story 4 to the list of movies (you can use any director)

SQL Query: **INSERT** **INTO** movies **values(**14**,**"Toy Story 4"**,**"John Lasseter"**,**2020**,**120**);**A screenshot of a computer

Description automatically generated

1. 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.

SQL Query: **insert** **into** boxoffice **values(**14**,**8.7**,**340000000**,**270000000**);**

**Updating rows**

**Exercise 14 – Tasks**

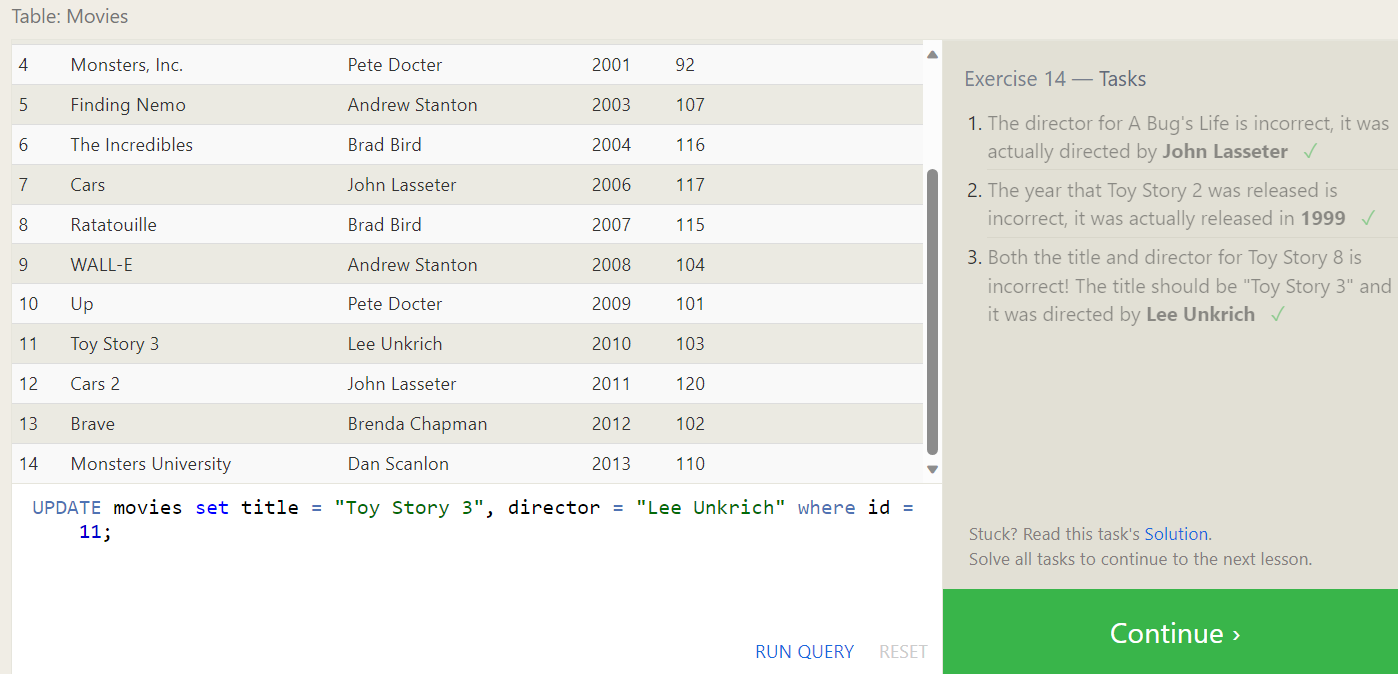
1. The director for A Bug's Life is incorrect, it was actually directed by John Lasseter.

SQL Query: **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

SQL Query: **UPDATE** movies **set** year **=** 1999 **where** title **=** "Toy Story 2"**;**

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

SQL Query: **UPDATE** movies **set** title **=** "Toy Story 3"**,** director **=** "Lee Unkrich" **where** id **=** 11**;**

**Deleting rows**

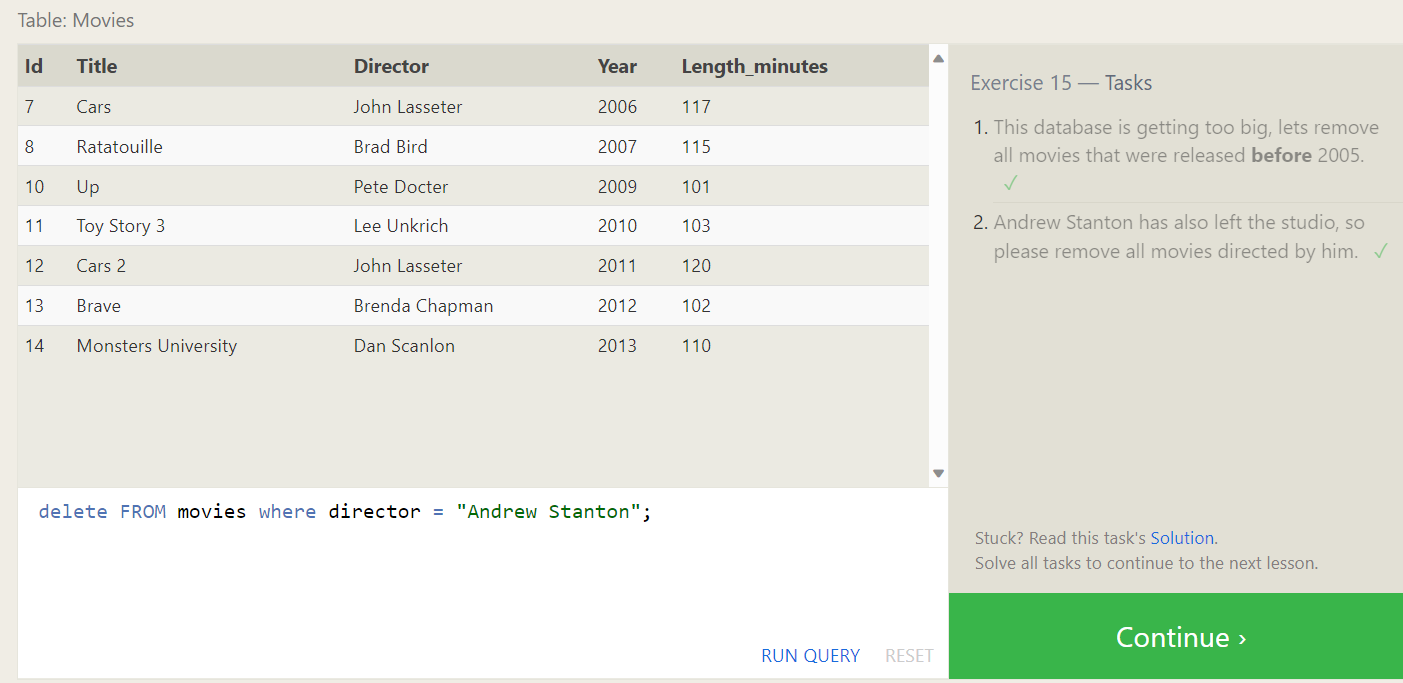
**Exercise 15 – Tasks**

1. This database is getting too big, lets remove all movies that were released before 2005.

SQL Query: **delete** **FROM** movies **where** year **<** 2005**;**A screenshot of a computer

Description automatically generated

1. Andrew Stanton has also left the studio, so please remove all movies directed by him.

SQL Query: **delete** **FROM** movies **where** director **=** "Andrew Stanton"**;**

**Creating tables**

**Exercise 16 – Tasks**

1. Create a new table named Database with the following columns:
   1. – Name A string (text) describing the name of the database
   2. – Version A number (floating point) of the latest version of this database
   3. – Download\_count An integer count of the number of times this database was downloaded
   4. This table has no constraints.

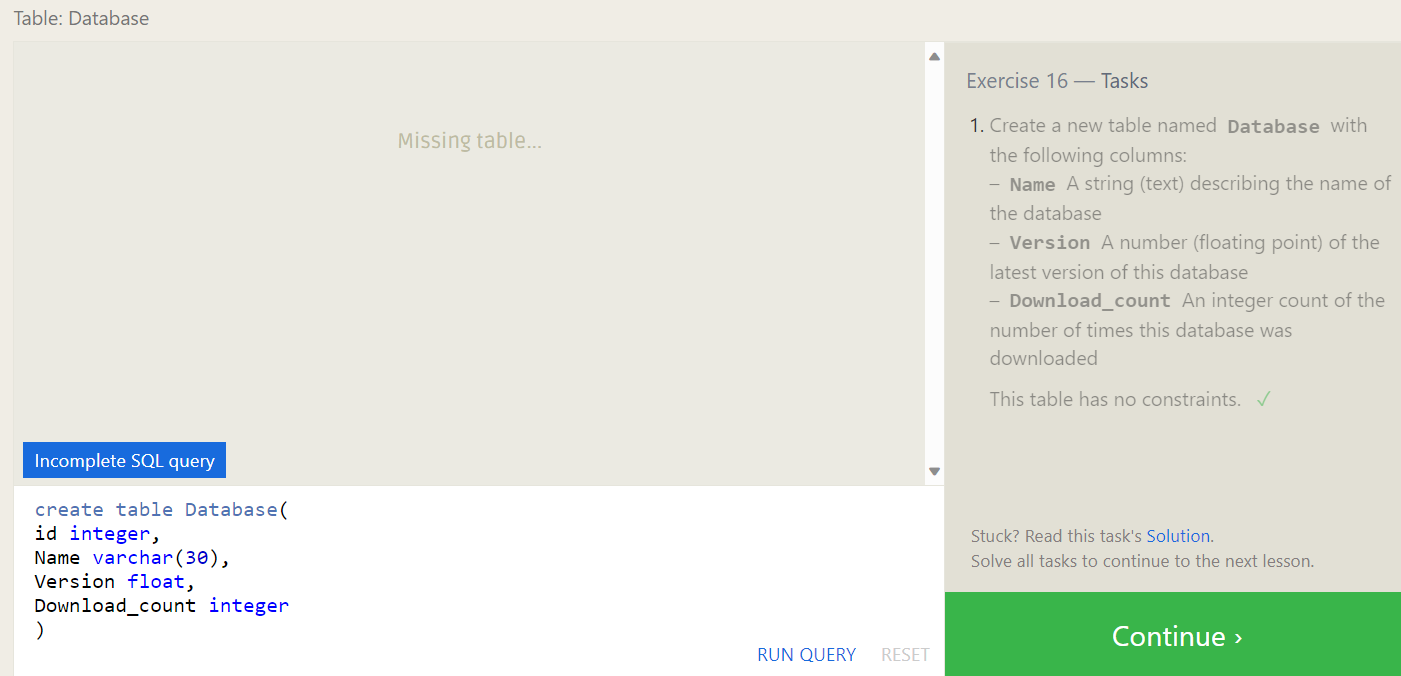
SQL Query: **create** **table** **Database(**

id integer**,**

Name varchar**(**30**),**

Version float**,**

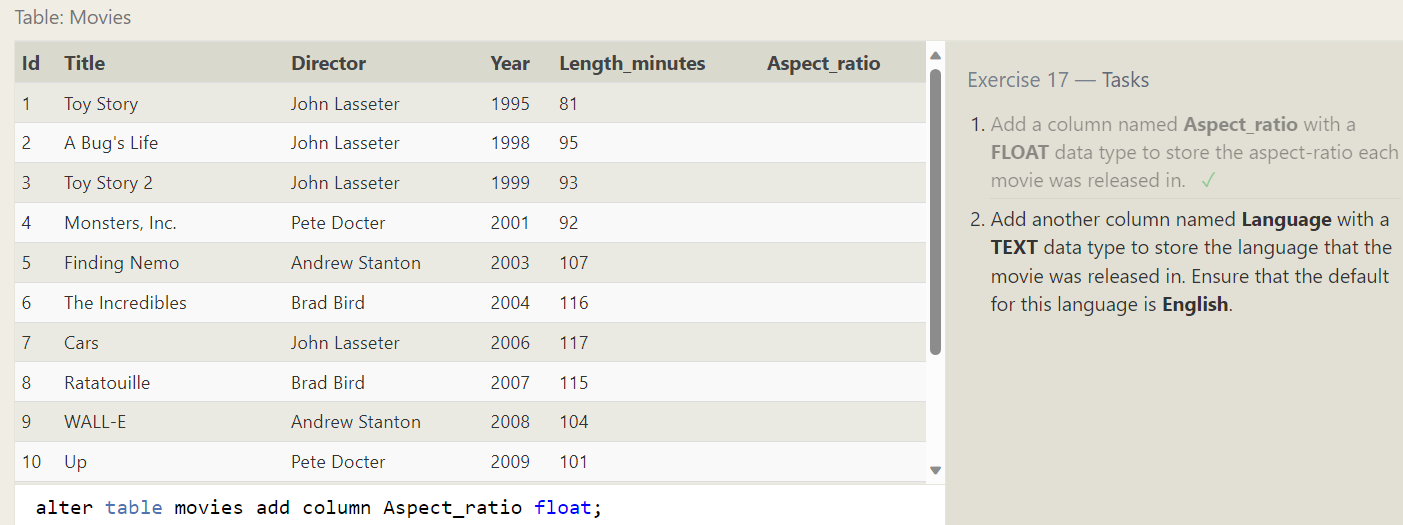
Download\_count integer

**)**

**Altering tables**

**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.

SQL Query: **alter** **table** movies **add** **column** 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.

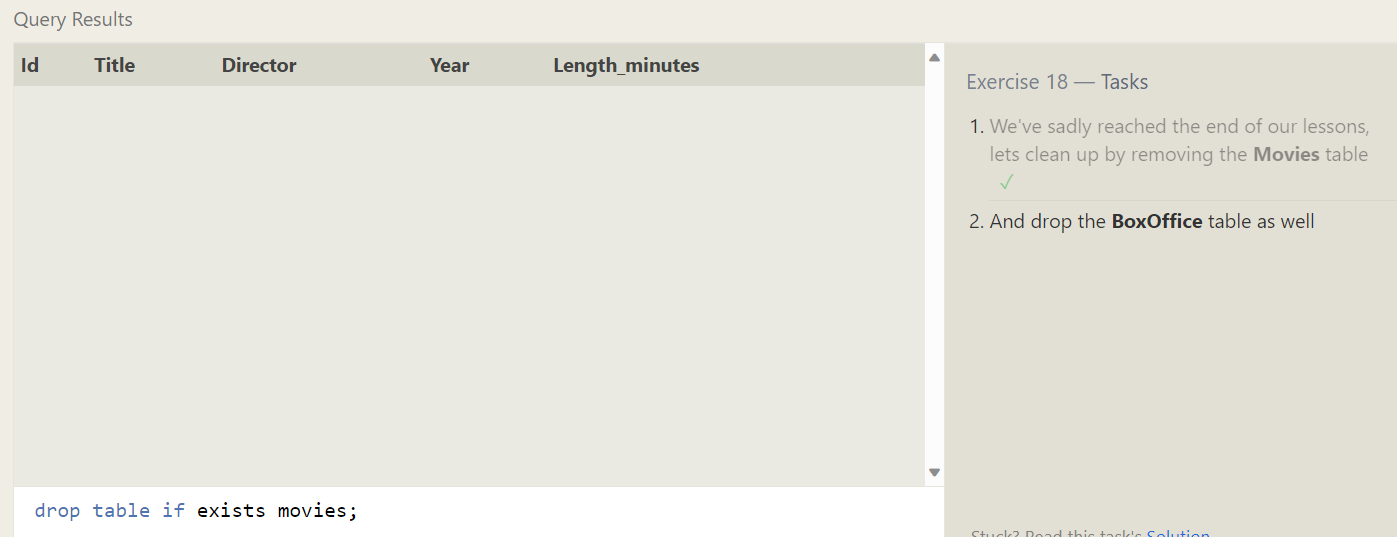
SQL Query: **alter** **table** movies **add** **column** **Language** text **default** English**;**A screenshot of a computer

Description automatically generated

**Dropping tables**

**Exercise 18 – Tasks**

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

SQL Query: **drop** **table** **if** **exists** movies**;**

1. And drop the BoxOffice table as well

SQL Query: **drop** **table** **if** **exists** boxoffice**;**A screenshot of a computer

Description automatically generated