**SQL exercise**

**SQL Lesson 1: SELECT queries 101**

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 (Pt. 1)**

1.Find the movie with a row id of 6

select \* from movies where id = 6

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

select \*from movies where year between 2000 and 2010

3.Find the movies not released in the years 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 \* from movies limit 5



**SQL Lesson 3: Queries with constraints (Pt. 2)**

1.Find all the Toy Story movies

select \* from movies where title like "Toy Story%"

2.Find all the movies directed by John Lasseter

select \* from movies where director="John Lasseter";

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

select \* from movies where director!="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 \* from movies order by year desc limit 4

3.List the first five Pixar movies sorted alphabetically

select \* from movies order by title limit 5

4.List the next five Pixar movies sorted alphabetically

select \* from movies order by title limit 5 offset 5



**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 \* from north\_american\_cities where longitude <-87.629798

ORDER BY longitude

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

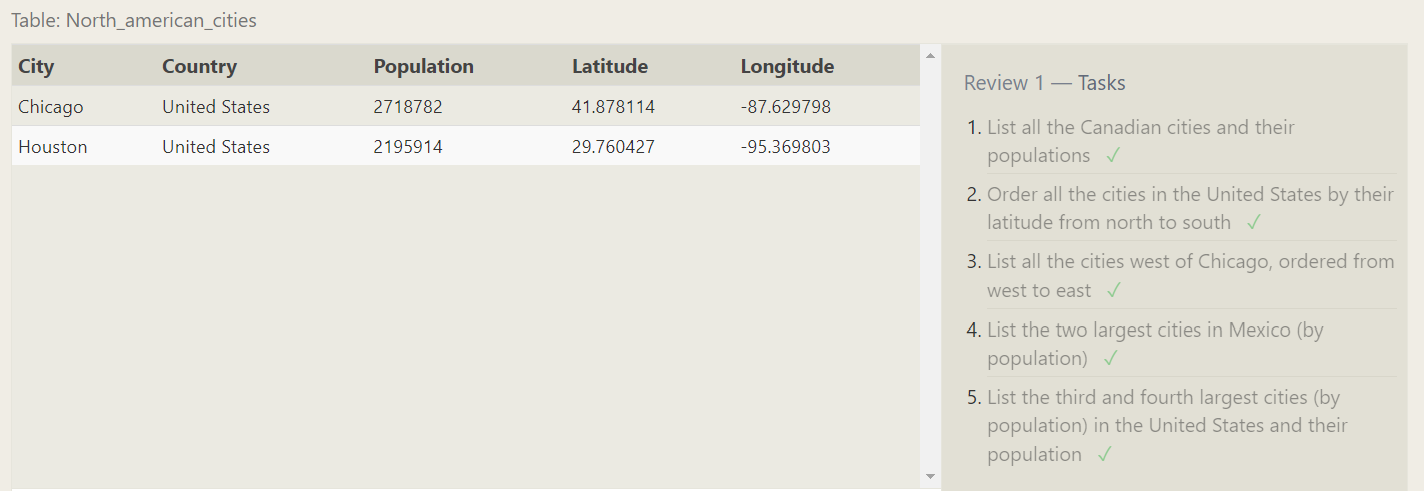
select \* from north\_american\_cities where country ='Mexico'

ORDER BY population desc limit 2

5.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



**SQL Lesson 6: Multi-table queries with JOINs**

1.Find the domestic and international sales for each movie

select \* from Movies m join Boxoffice b on m.id = b.movie\_id

2.Show the sales numbers for each movie that did better internationally rather than domestically

select \* from Movies m join boxoffice b on m.id = b.Movie\_id where International\_sales > Domestic\_sales

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

select \* from Movies m join boxoffice b on m.id = b.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



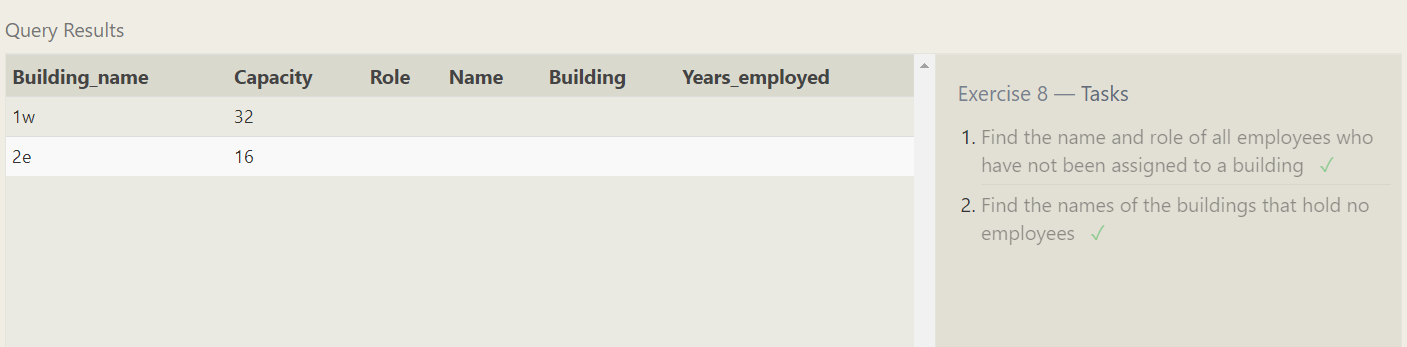
**SQL Lesson 8: A short note on NULLs**

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 \* from Buildings left join Employees on Building\_name = Building where Name 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 total\_sale from movies m join boxoffice b on m.id = b.movie\_id

2.List all movies and their ratings in percent

select title, (rating)\*10 from movies m join boxoffice b on m.id = b.movie\_id

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

select title from movies m join boxoffice b on m.id = b.movie\_id

WHERE year%2=0



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

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

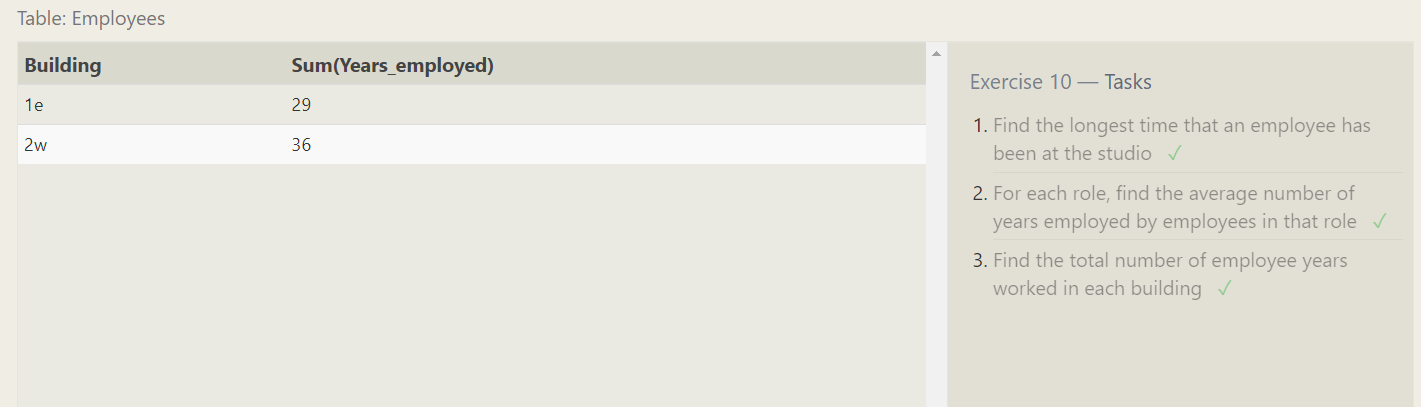
select max(years\_employed) from employees;

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

select role, avg(years\_employed) from employees group by role;

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

select building, sum(years\_employed) from employees group by building



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

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

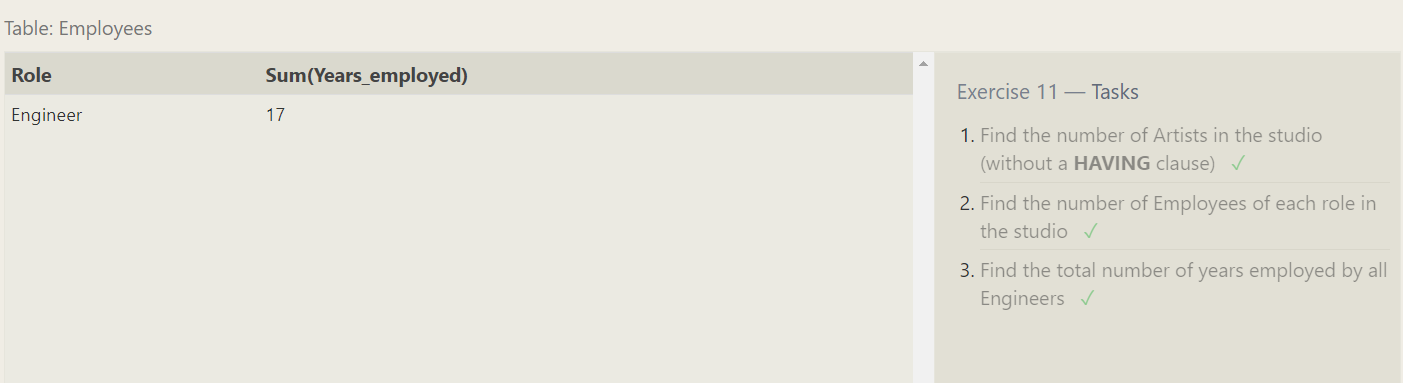
SELECT count(name) FROM employees where role='Artist'

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

select role,count(name) 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'



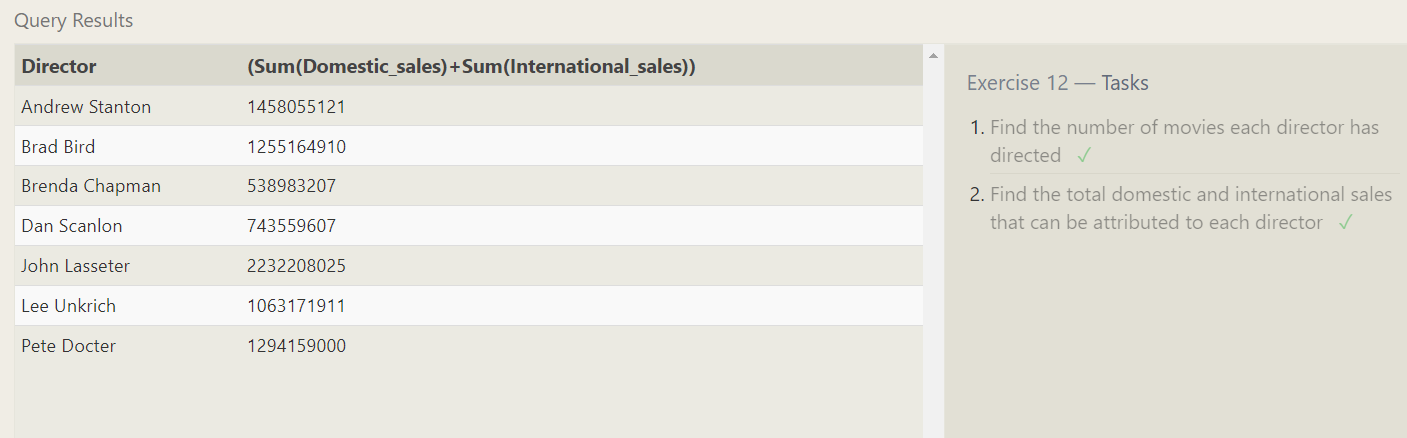
**SQL Lesson 12: Order of execution of a Query**

1.Find the number of movies each director has directed

select Director,count(title) from Movies m join boxoffice b on m.id = b.movie\_id group by Director;

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

select director,(sum(Domestic\_sales)+sum(International\_sales)) from Movies m join boxoffice b on m.id = b.movie\_id group by Director



**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)

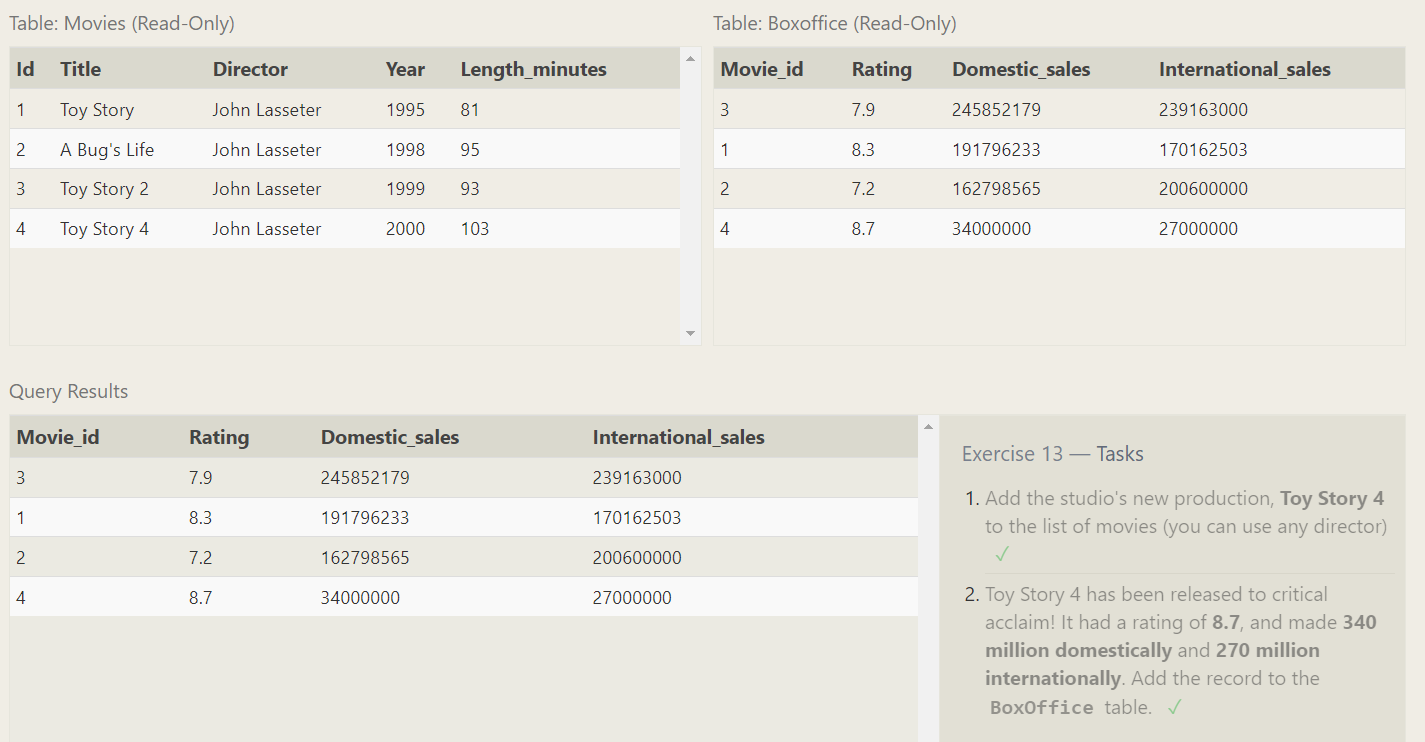
insert into Movies values

(4,'Toy Story 4','John Lasseter',2000,103);

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.

insert into Boxoffice values

(4,8.7,34000000,27000000);



**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 Title = "A Bug's Life";

2. The year that Toy Story 2 was released is incorrect, it was actually released in 1999

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

update Movies

set

Title = 'Toy Story 3',

Director = 'Lee Unkrich'

Where

Title = 'Toy Story 8'



**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.

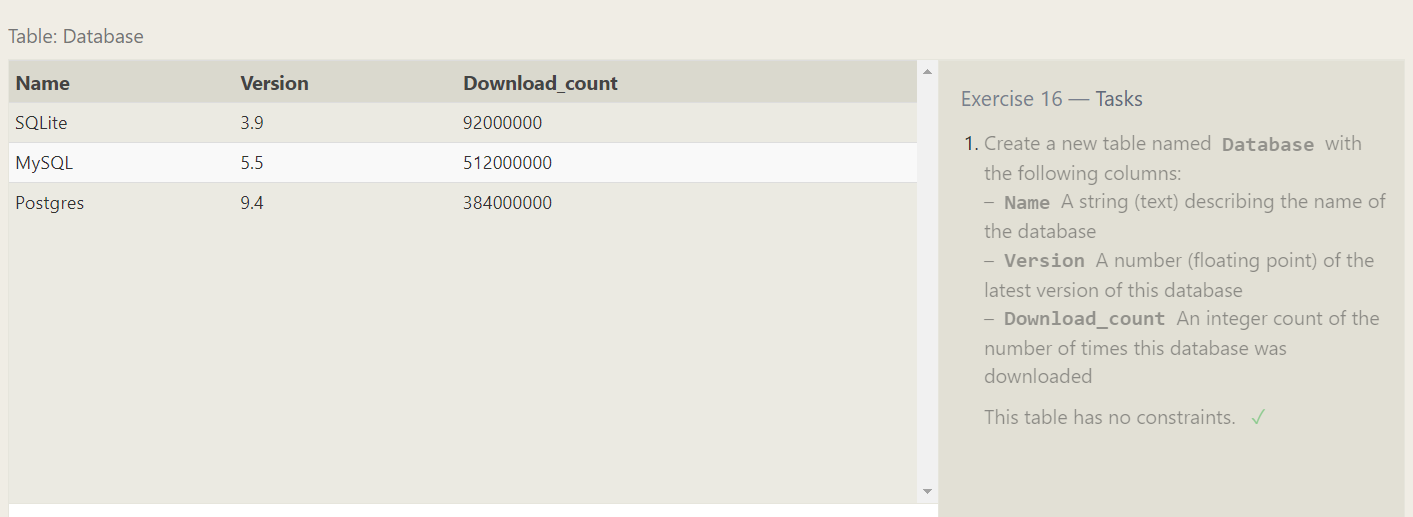
create table Database (

Name varchar,

Version float,

Download\_count int

);



**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.

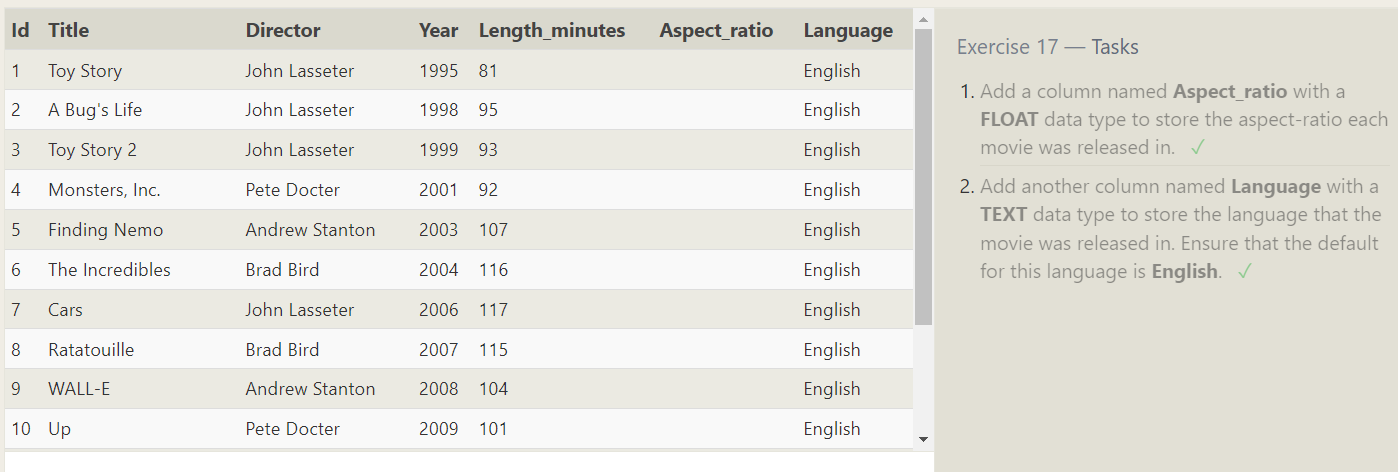
alter table Movies

add Aspect\_ratio float;

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 Language varchar(255) default ('English');



**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;

