**SQL Lesson 1: SELECT queries 101**

1. **Find the title of each film**



1. **Find the director of each film**



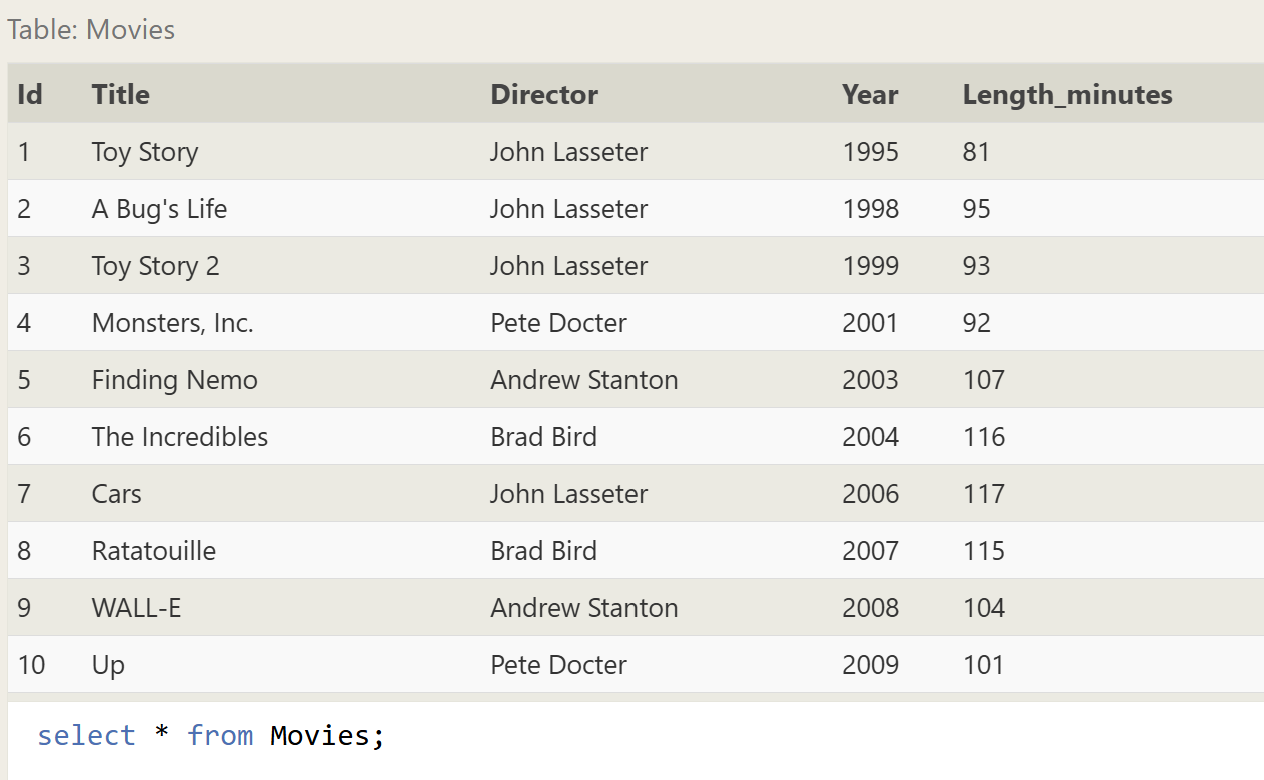
1. **Find the title and director of each film**

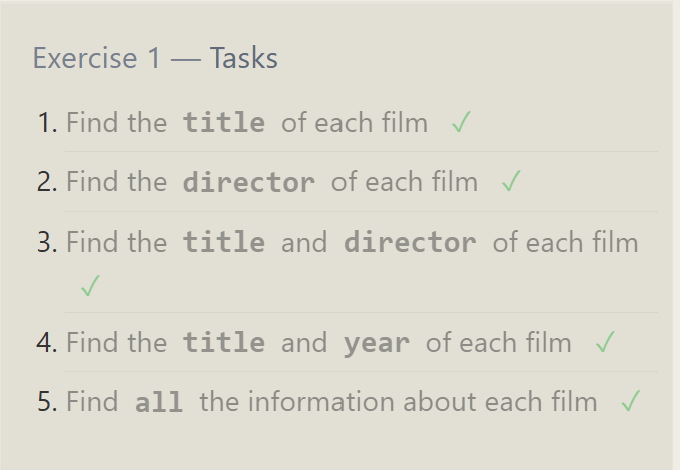


1. **Find the title and year of each film**

****

1. **Find all the information about each film**

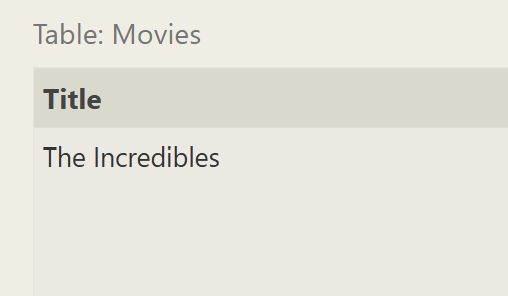
****

****

**SQL Lesson 2: Queries with constraints (Pt. 1)**

1. **Find the movie with a row id of 6**

SELECT title from Movies where id = 6;



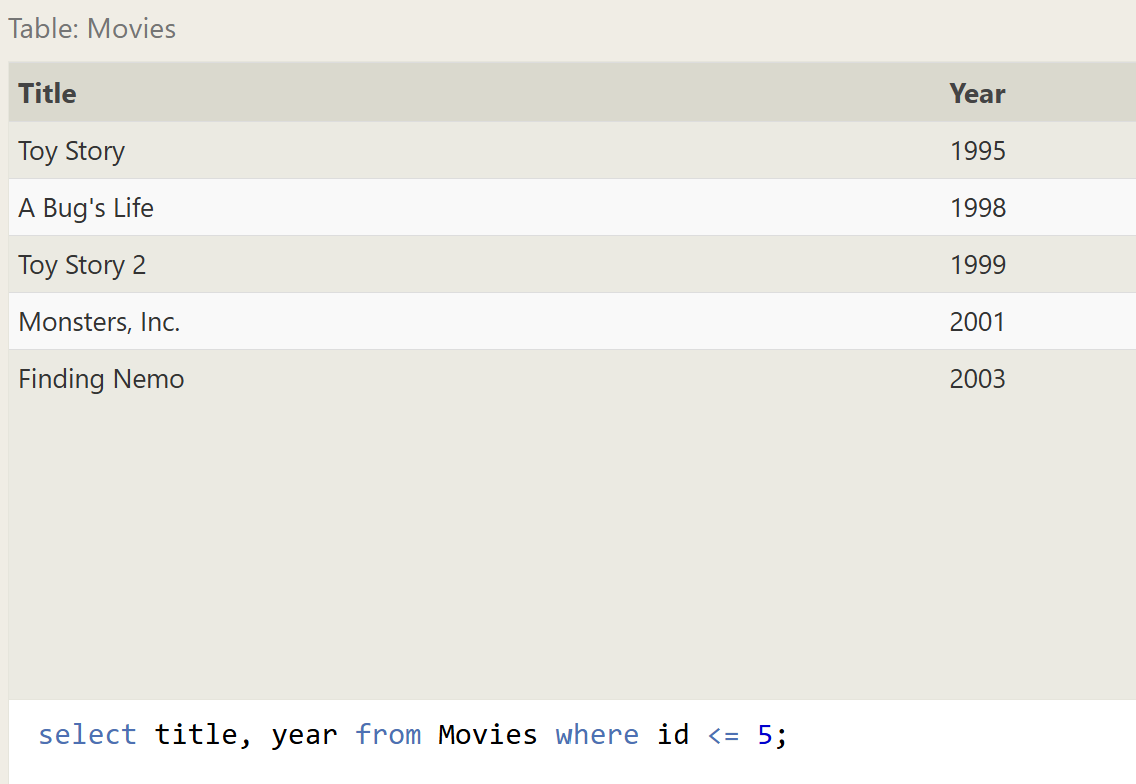
1. **Find the movies released in the years between 2000 and 2010**

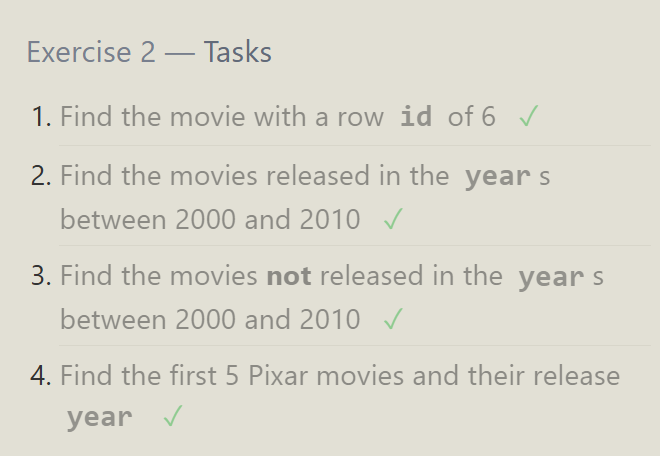


1. **Find the movies not released in the years between 2000 and 2010**



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

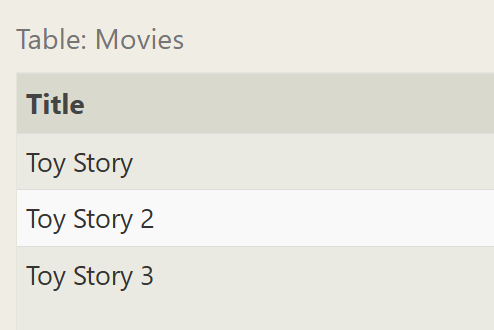
****

****

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

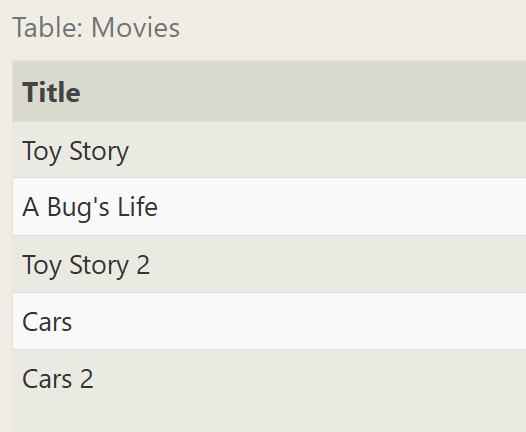
1. **Find all the Toy Story movies**

select title from Movies where title like 'Toy Story%'

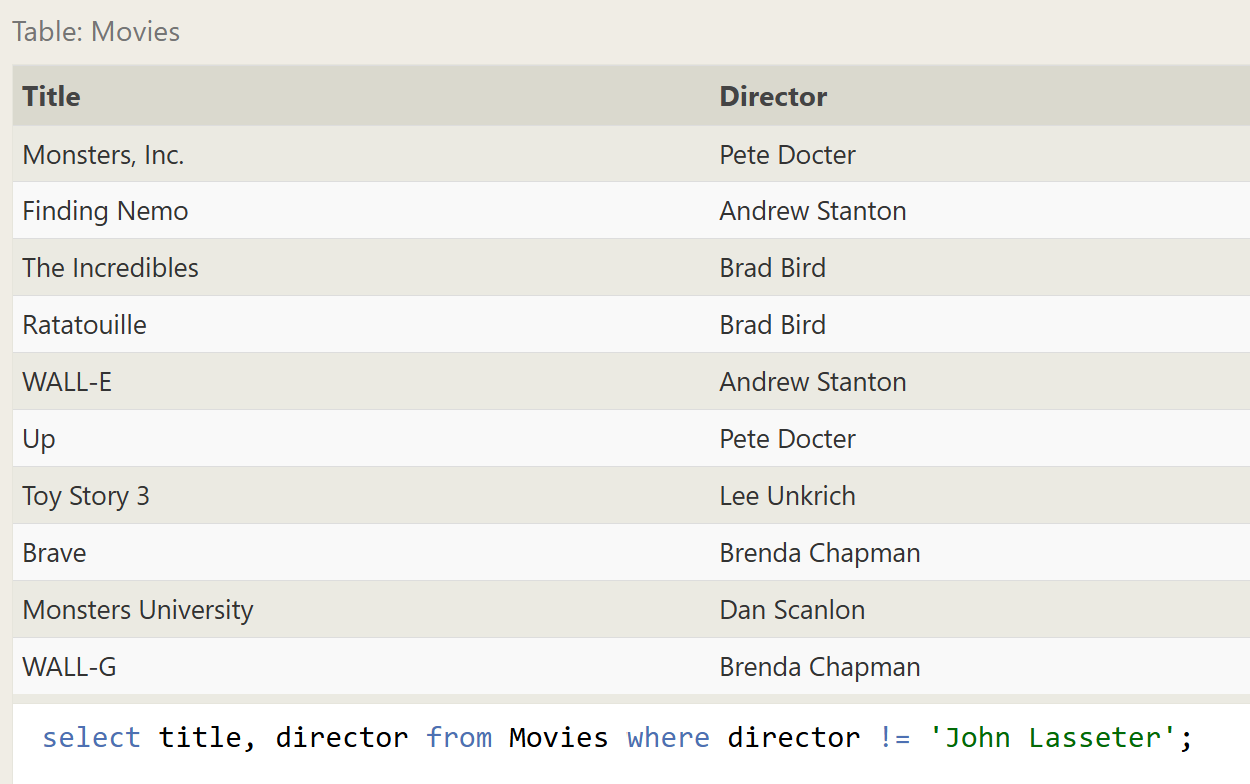


1. **Find all the movies directed by John Lasseter**

select title from Movies where Director = 'John Lasseter';

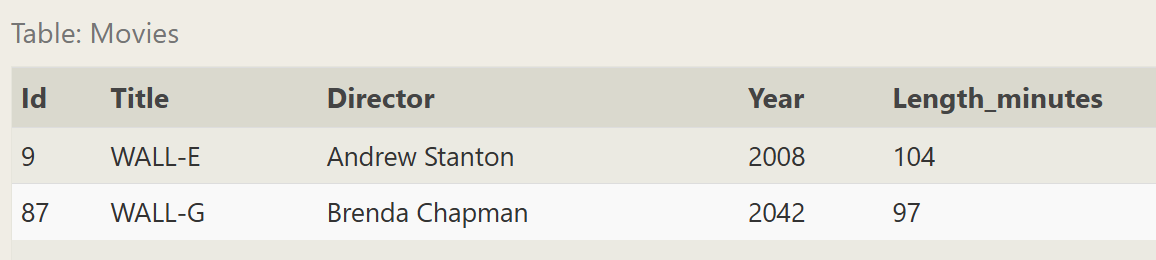


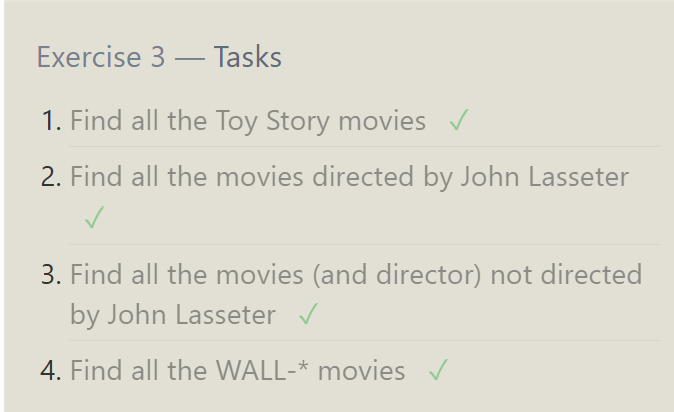
1. **Find all the movies (and director) not directed by John Lasseter**

****

1. **Find all the WALL-\* movies**

select title from Movies where title like "WALL-\_"





**SQL Lesson 4: Filtering and sorting Query results**

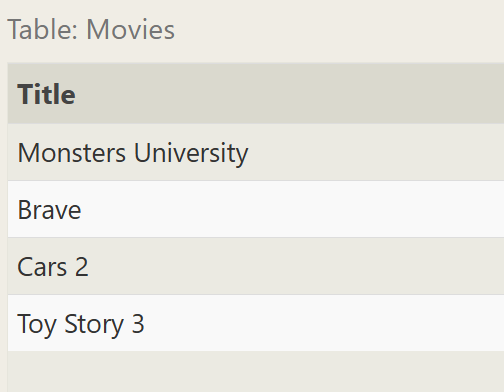
1. **List all directors of Pixar movies (alphabetically), without duplicates**

SELECT distinct Director FROM movies order by director;



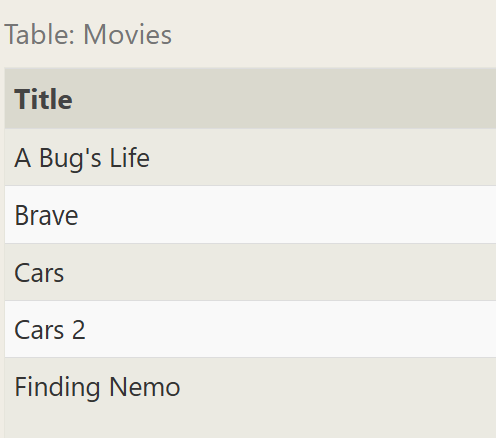
1. **List the last four Pixar movies released (ordered from most recent to least)**

select title from Movies order by year desc limit 4;

****

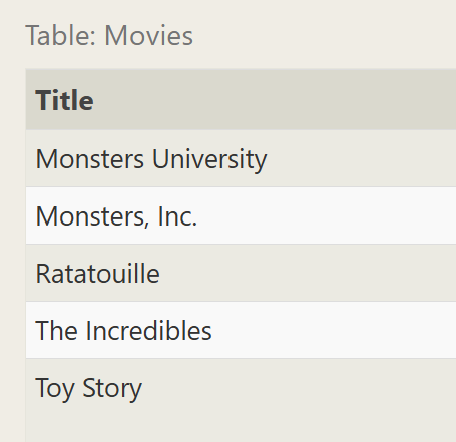
1. **List the first five Pixar movies sorted alphabetically**

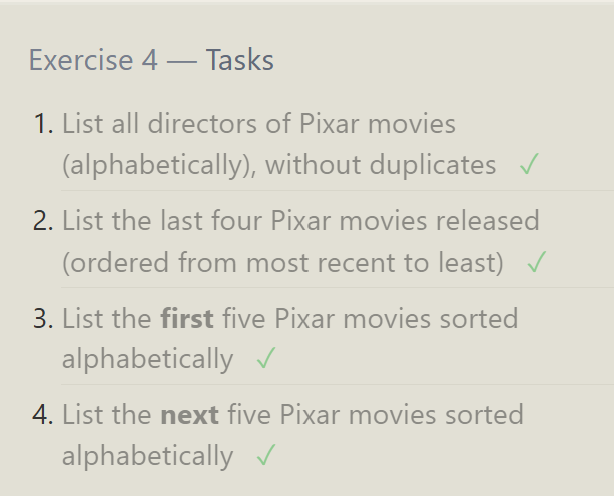
select \* from Movies order by title limit 5;

****

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

select title from Movies order by title limit 5 offset 5;





**SQL Review: Simple SELECT Queries**

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 \* FROM north\_american\_cities where country = "United States" order by latitude desc;



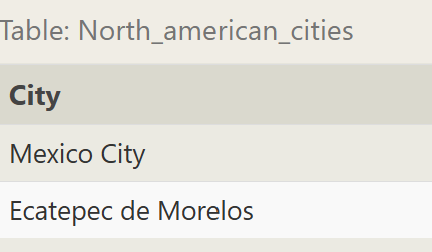
1. **List all the cities west of Chicago, ordered from west to east**

SELECT city FROM north\_american\_cities where longitude < -87.629798 order by longitude ASC;



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

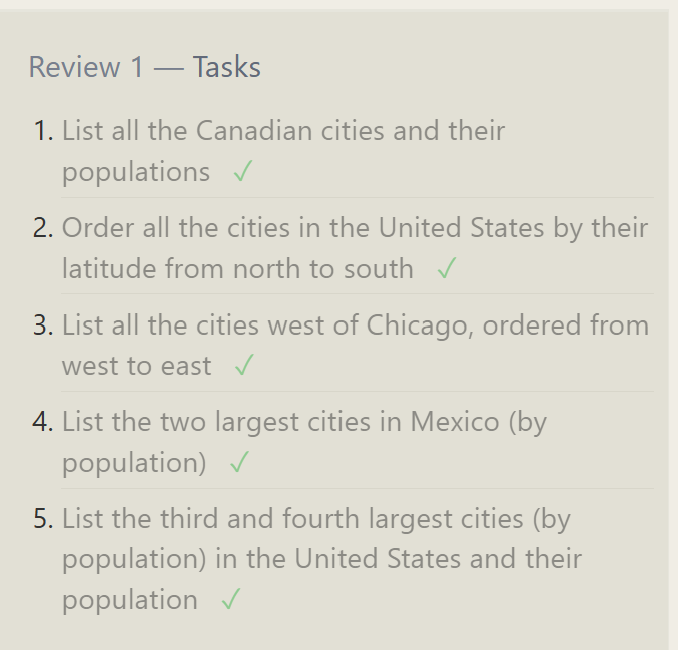
select city from north\_american\_cities where country = "Mexico" order by population desc limit 2;



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

select city, population from north\_american\_cities where country = "United States" order by population desc limit 2 offset 2;

****

****

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

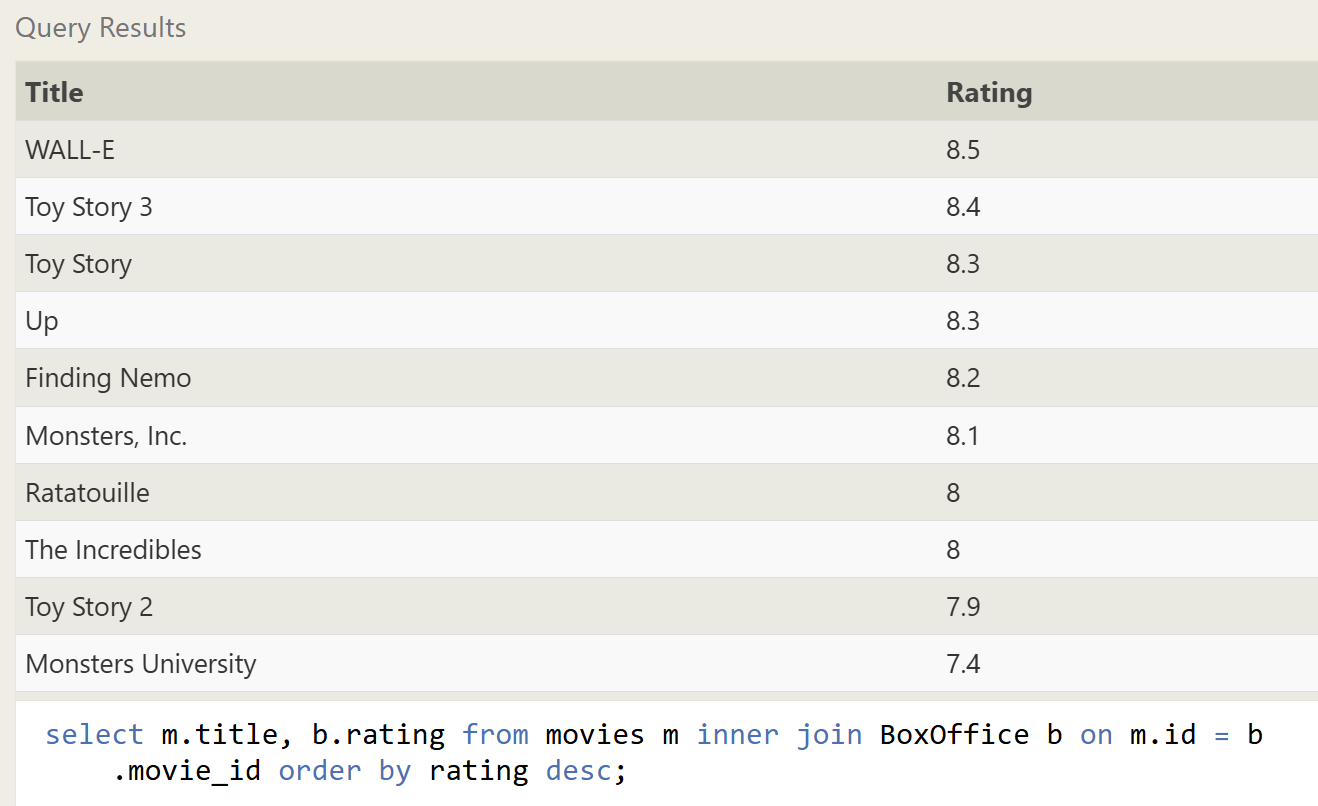
1. **Find the domestic and international sales for each movie**

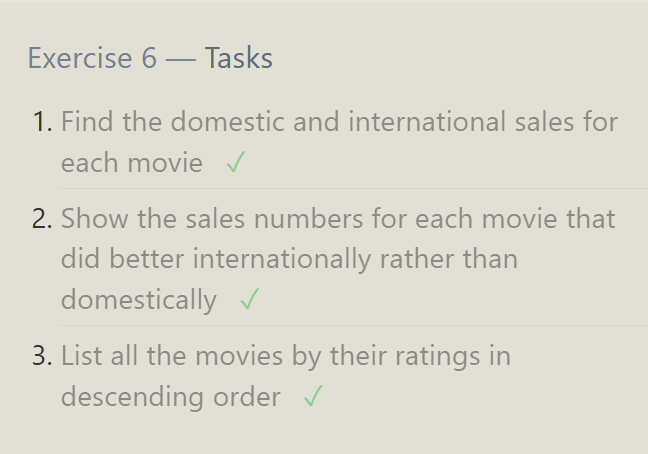
****

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

****

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

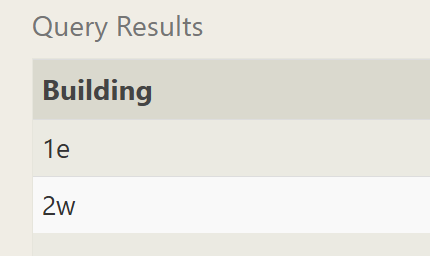
****

****

**SQL Lesson 7: OUTER JOINs**

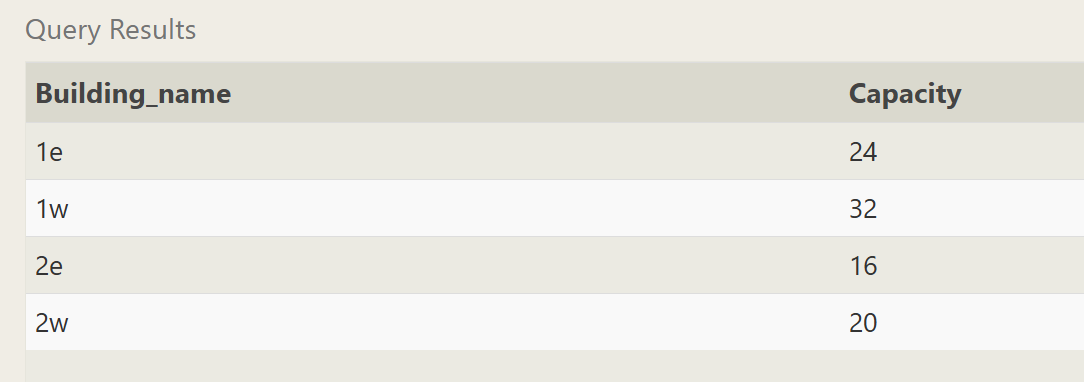
1. **Find the list of all buildings that have employees**

select distinct building from Employees;



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

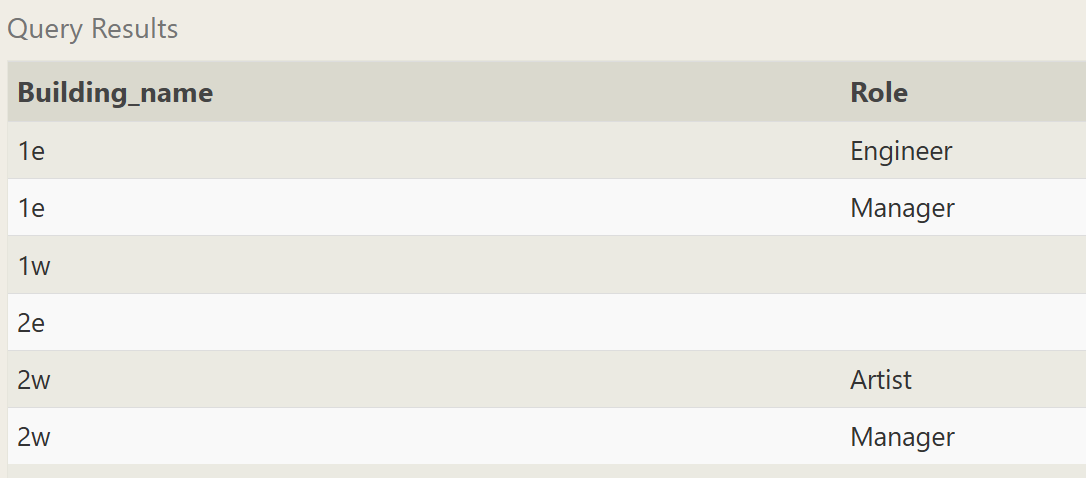
select \* from buildings;

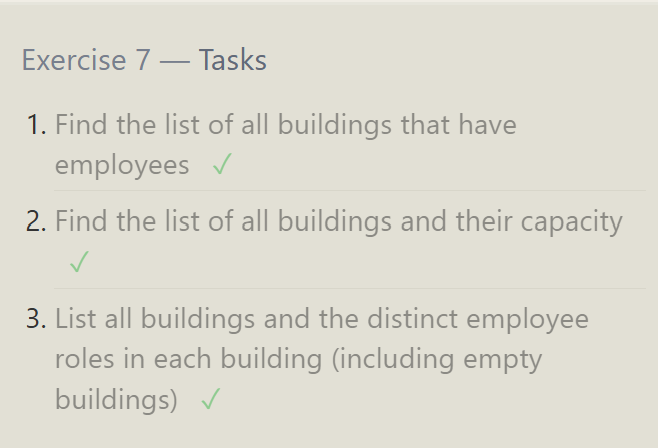


1. **List all buildings and the distinct employee roles in each building (including empty buildings)**

SELECT DISTINCT building\_name, role FROM buildings LEFT JOIN employees

ON building\_name = building

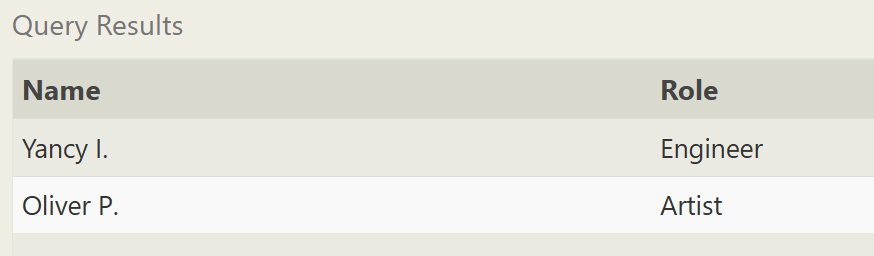




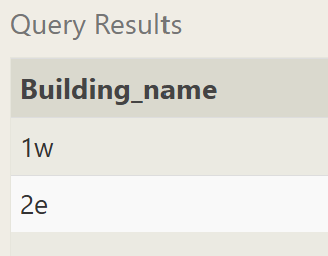
**SQL Lesson 8: A short note on NULLs**

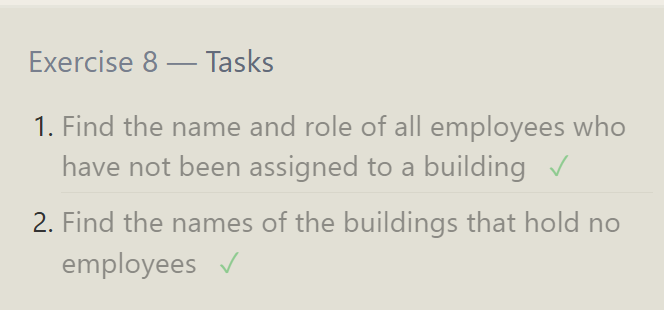
1. **Find the name and role of all employees who have not been assigned to a building**

SELECT name, role FROM employees where building is NULL;



1. **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;

****

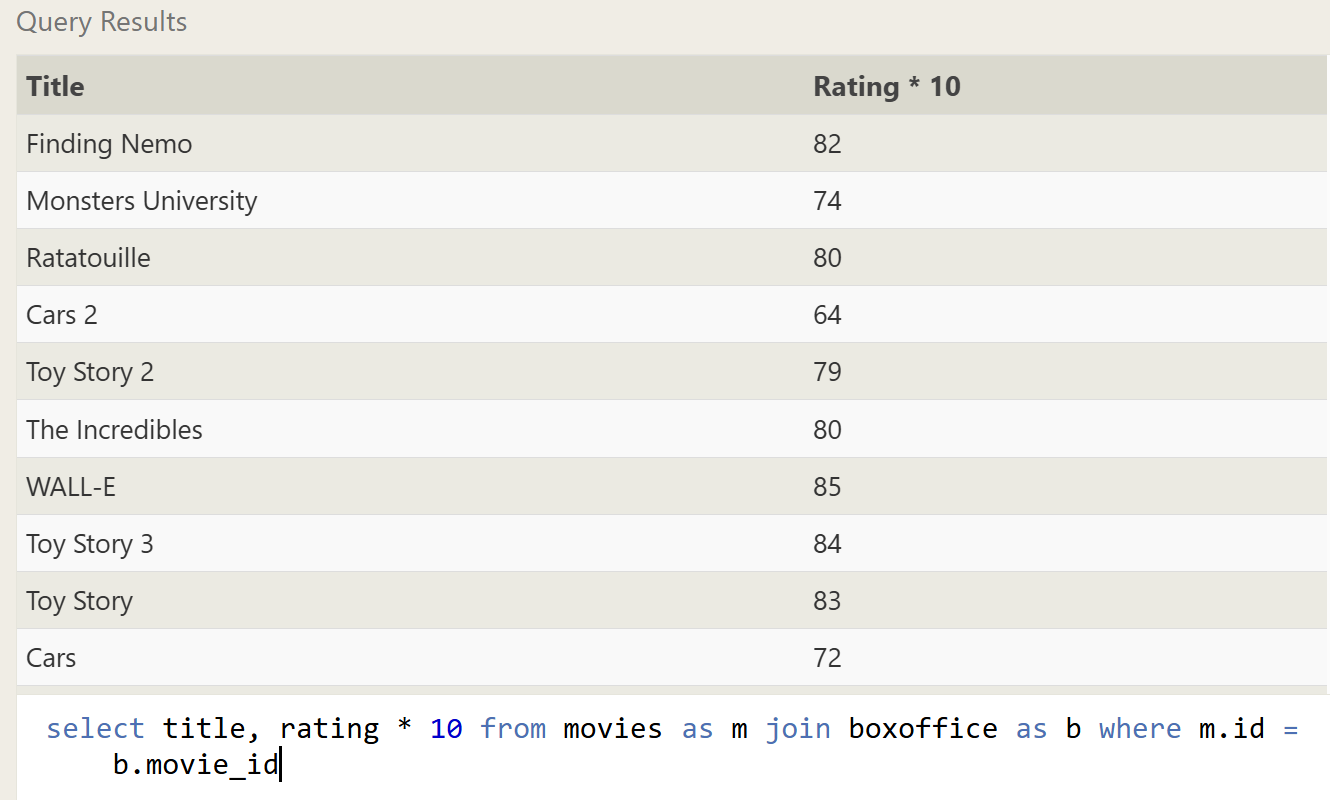
****

**SQL Lesson 9: Queries with expressions**

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



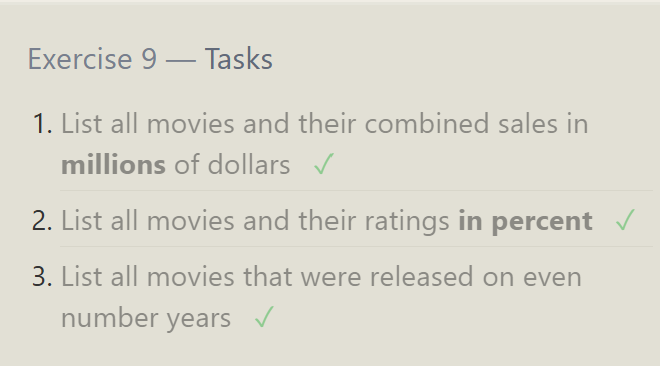
1. **List all movies and their ratings in percent**



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

select \* from movies where year % 2 = 0;

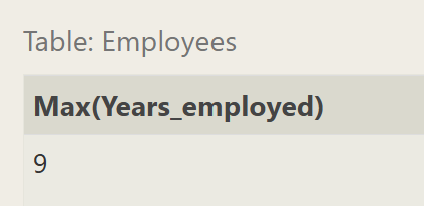
****

****

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

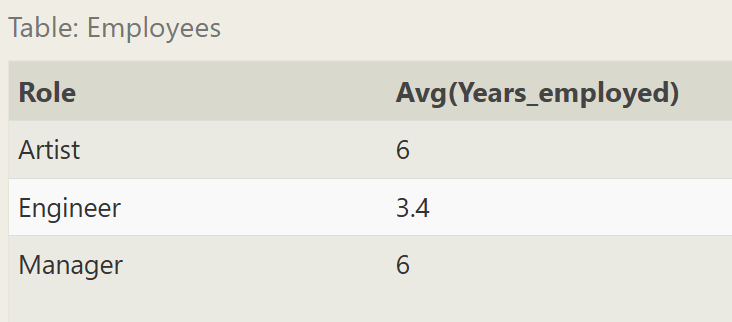
1. **Find the longest time that an employee has been at the studio**

SELECT max(Years\_employed) FROM employees;

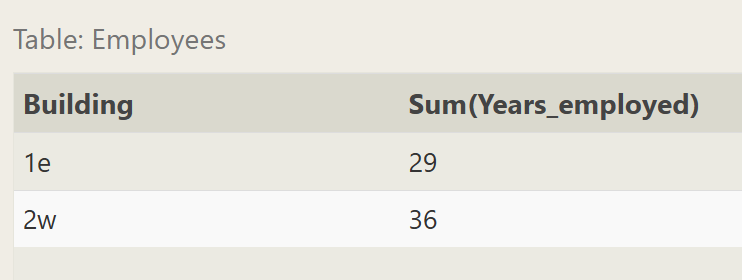


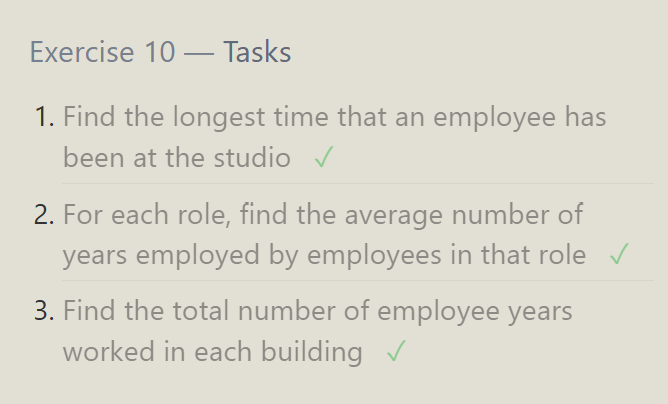
1. **For each role, find the average number of years employed by employees in that role**

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



1. **Find the total number of employee years worked in each building**select building, sum(Years\_employed) from employees group by building;

****

****

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

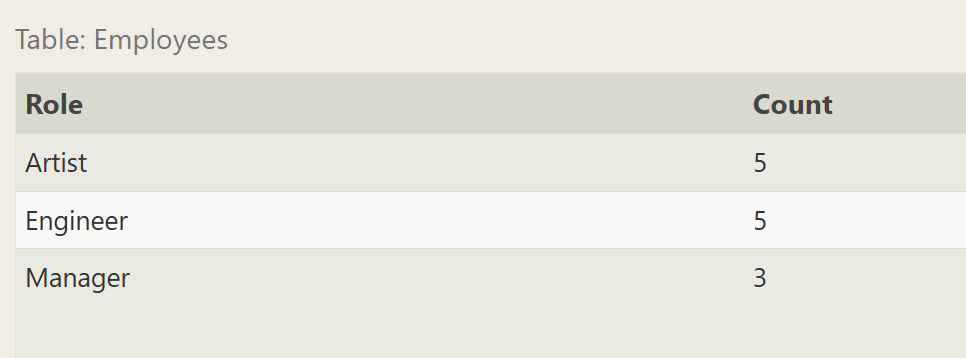
1. **Find the number of Artists in the studio (without a HAVING clause)**

SELECT role, count(\*) as Count FROM employees where Role = "Artist";



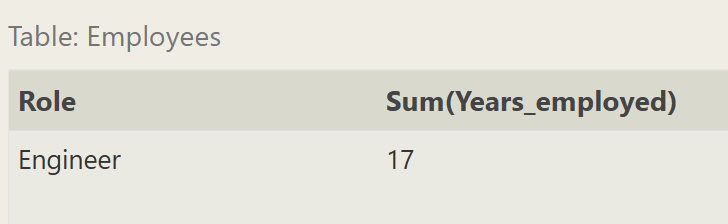
1. **Find the number of Employees of each role in the studio**

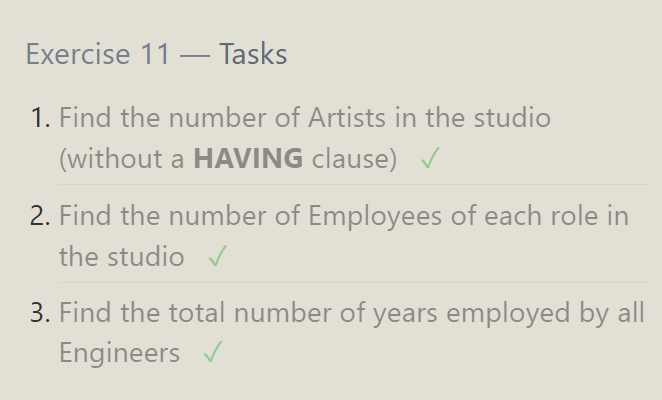
select role, count(\*) as Count from Employees group by role;



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

select role, sum(Years\_employed) from employees group by role having role = "Engineer";

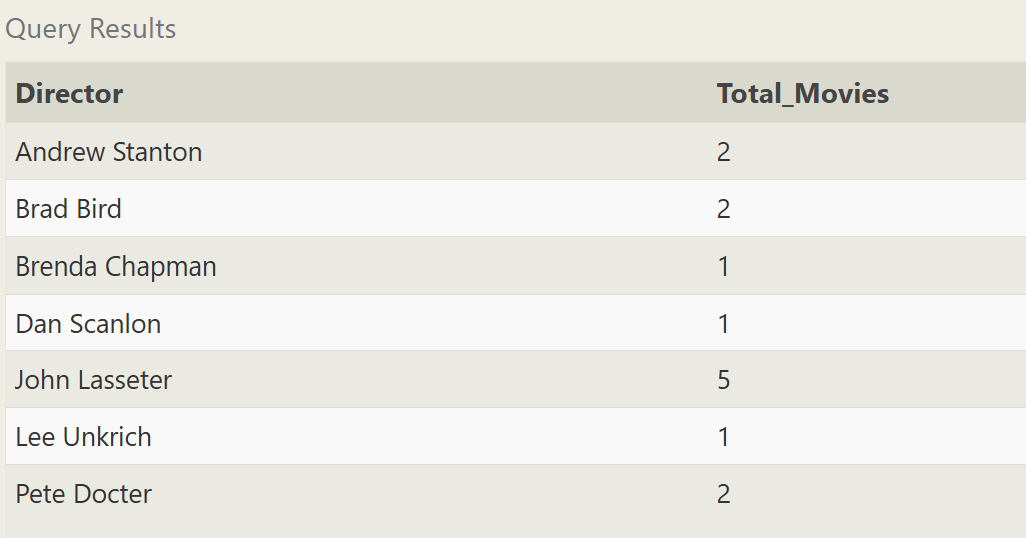




**SQL Lesson 12: Order of execution of a Query**

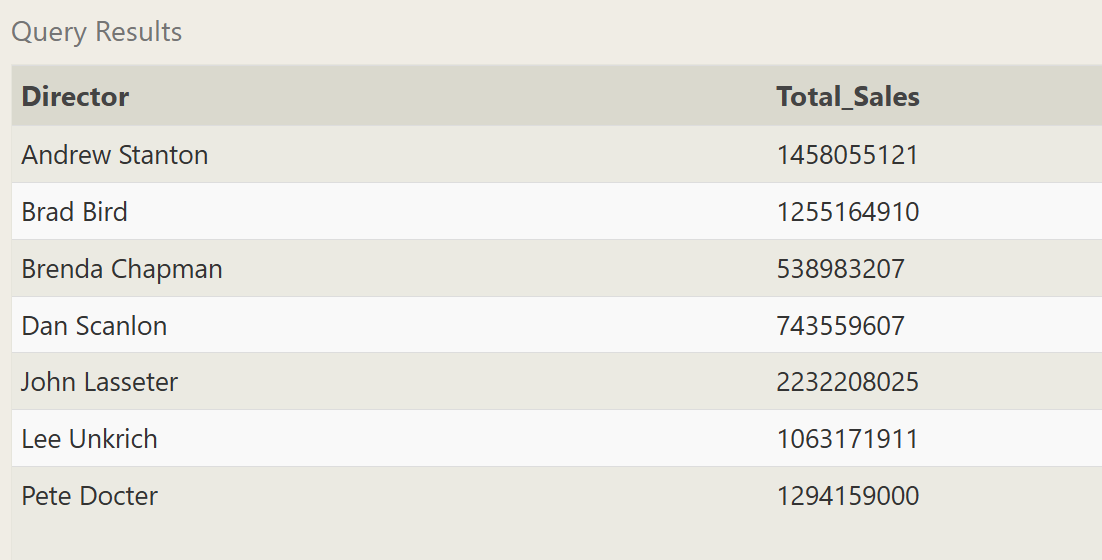
1. **Find the number of movies each director has directed**

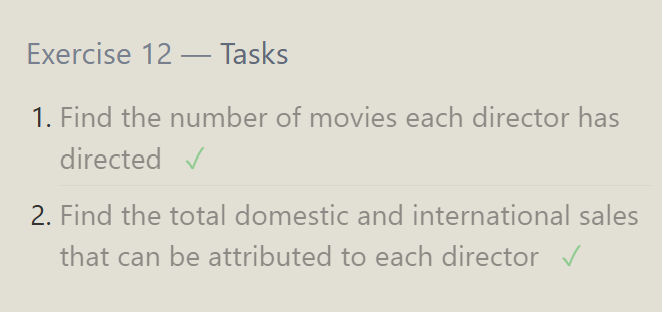
SELECT director, count(\*) as Total\_Movies FROM movies group by director;



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 as m join boxoffice b where m.id = b.movie\_id group by director;





**SQL Lesson 13: Inserting rows**

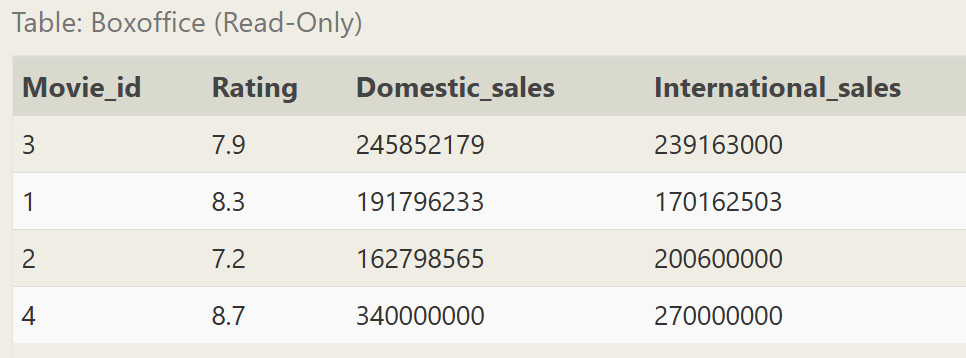
1. **Add the studio's new production, Toy Story 4 to the list of movies (you can use any director)**

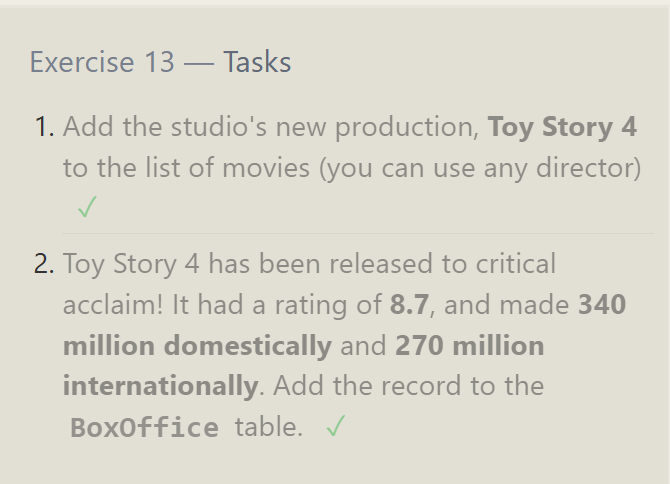
insert into Movies values(4, 'Toy Story 4', 'John Lasseter', 1996, 90)



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, 340000000, 270000000);





**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";



1. **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';



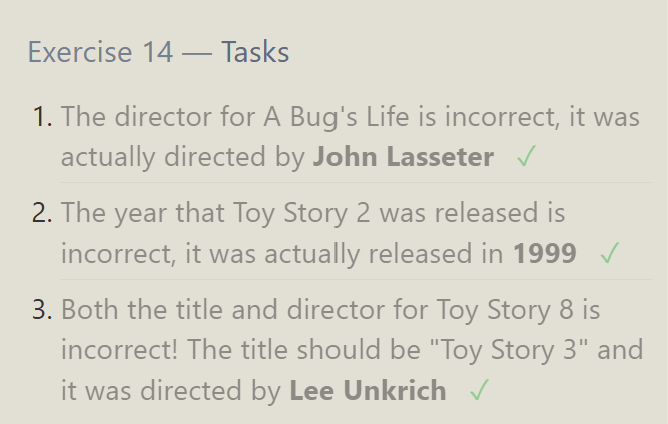
1. **Both the title and director for Toy Story 8 is incorrect! The title should be "Toy Story 3" and it was directed by Lee Unkrich**

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.**

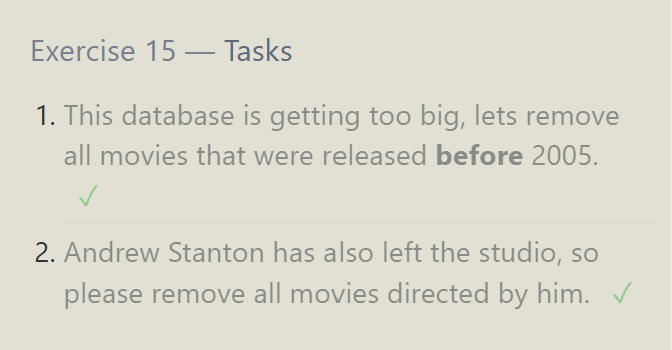
delete from Movies where Year < 2005;



1. **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**

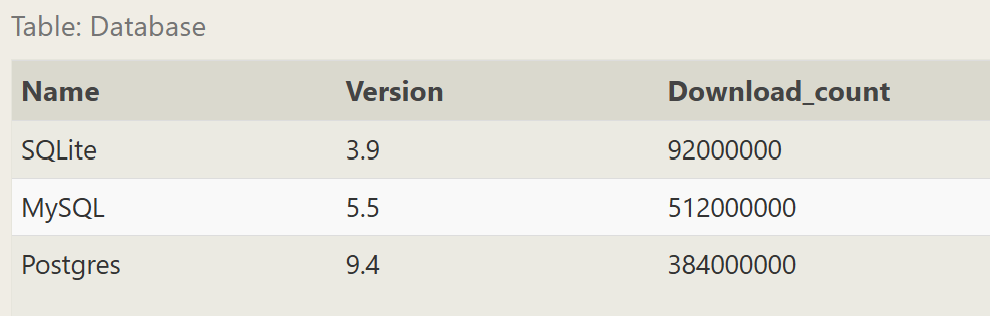
create table Database(

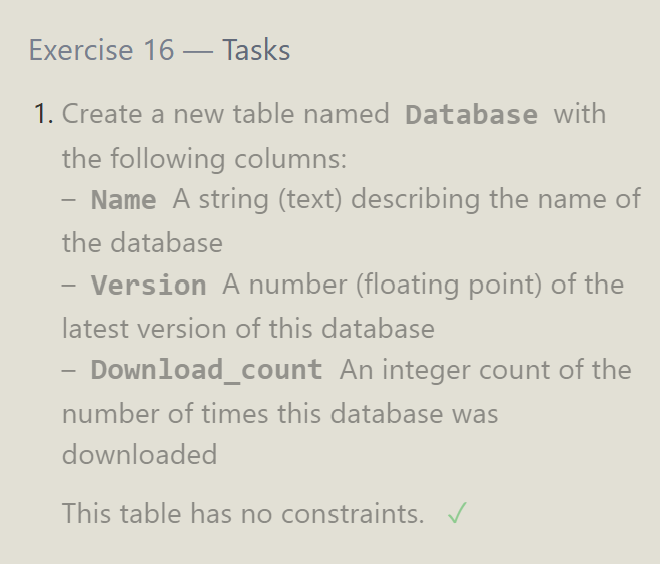
name varchar(255),

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

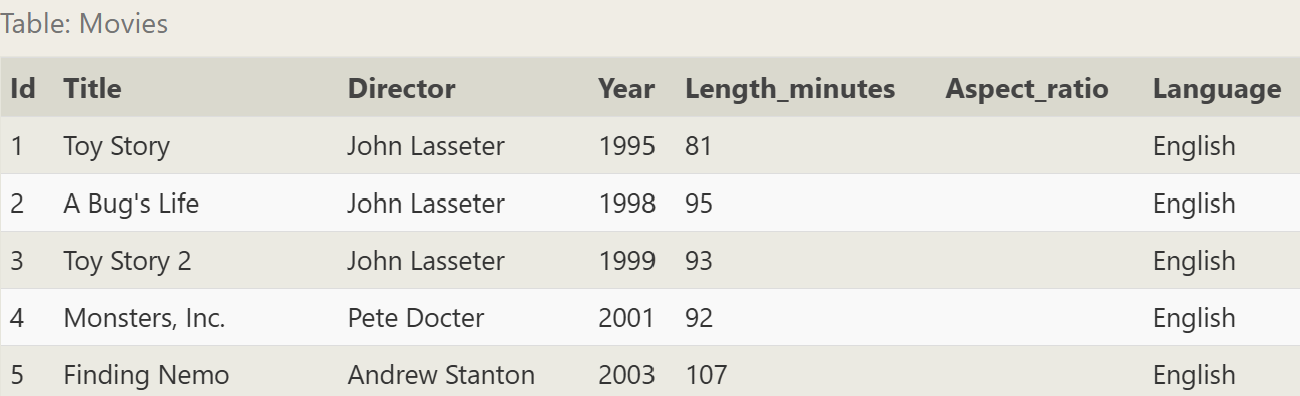
add Aspect\_ratio float;

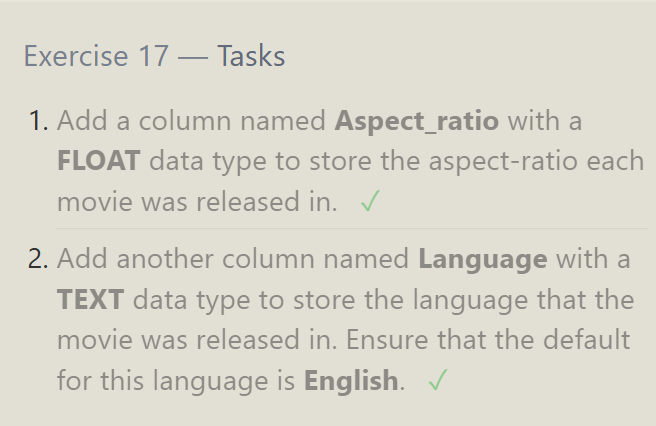


1. **Add another column named Language with a TEXT data type to store the language that the movie was released in. Ensure that the default for this language is English.**

alter table Movies

add Language text 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**

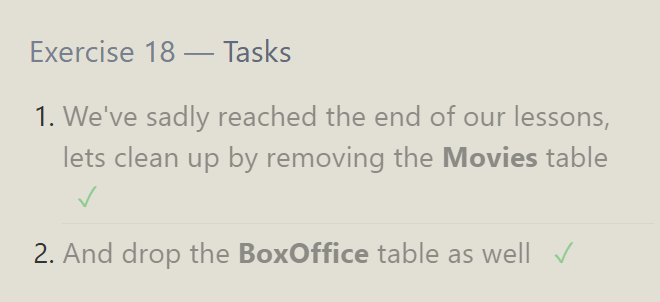
drop table Movies;



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

drop table BoxOffice;

****

****