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;

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 **year**s between 2000 and 2010

SELECT \* FROM movies where year between 2000 and 2010;

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

SELECT \* FROM movies where year not in between 2000 and 2010;

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

SELECT \* FROM movies limit 5;

Exercise 3 — Tasks

1. Find all the Toy Story movies

SELECT \* FROM movies where title like '%Toy Story%';

1. Find all the movies directed by John Lasseter

SELECT \* FROM movies where director = 'John Lasseter';

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

SELECT \* FROM movies where director != 'John Lasseter';

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;

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 < (SELECT longitude FROM north\_american\_cities WHERE city = 'Chicago') 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;

Exercise 6 — Tasks

1. Find the domestic and international sales for each movie

SELECT m.title, b.domestic\_sales, b.international\_sales FROM movies m JOIN boxoffice b ON m.id = b.movie\_id;

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

SELECT m.title, b.domestic\_sales, b.international\_sales FROM movies m JOIN boxoffice b ON m.id = b.movie\_id WHERE b.international\_sales > b.domestic\_sales;

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

SELECT m.title, b.rating FROM movies m JOIN boxoffice b ON m.id = b.movie\_id ORDER BY b.rating DESC;

Exercise 7 — Tasks

1. Find the list of all buildings that have employees

SELECT DISTINCT e.building FROM employees e;

1. Find the list of all buildings and their capacity

SELECT b.building\_name, b.capacity FROM buildings b;

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

SELECT b.building\_name, e.role FROM buildings b LEFT JOIN employees e ON b.building\_name = e.building GROUP 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 e.name, e.role FROM employees e WHERE e.building IS NULL;

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, b.rating \* 10 AS rating\_percent FROM movies m JOIN boxoffice b ON m.id = b.movie\_id;

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

SELECT title FROM movies 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 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(\*) FROM employees WHERE role = 'Artist';

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

SELECT role, COUNT(\*) FROM employees GROUP BY role;

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

SELECT SUM(years\_employed) FROM employees WHERE role = 'Engineer';

Exercise 12 — Tasks

1. Find the number of movies each director has directed

SELECT director, COUNT(\*) 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 cumulative\_sales\_from\_all\_movies 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 VALUES (15, 'Toy Story 4', 'John Lasseter', 2019, 100);

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 (15, 8.7, 340000000, 270000000);

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 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';

Exercise 15 — Tasks

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';

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 TEXT, Version REAL, 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 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**.

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;

