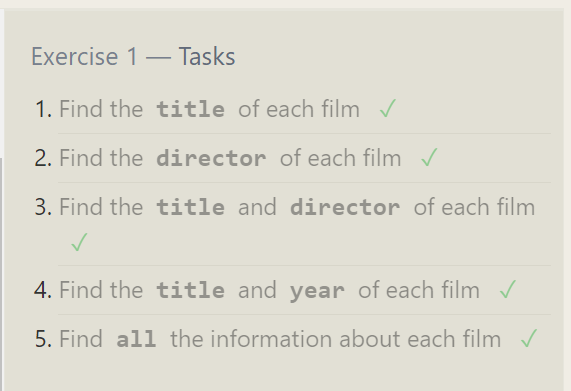
**SQL Lesson 1: SELECT queries 101**

1. SELECT title FROM movies;
2. SELECT director FROM movies;
3. SELECT title,director FROM movies;
4. SELECT title,year FROM movies;
5. SELECT \* FROM movies;

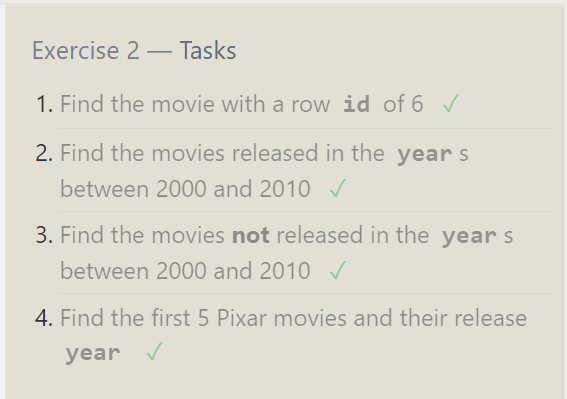
**Result:**

****

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

1. SELECT \* FROM movies where id=6;
2. SELECT \* FROM movies where year<=2010 and year>=2000;
3. SELECT \* FROM movies where year not between 2000 and 2010
4. SELECT title,year FROM movies where id<6

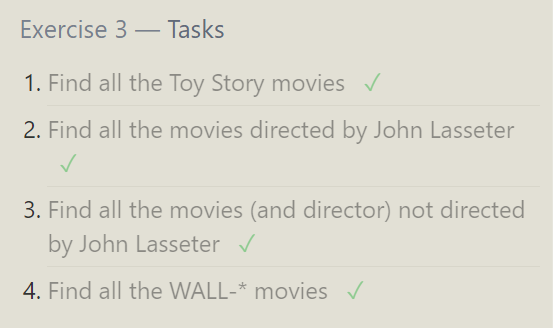
Result:

****

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

1. SELECT \* FROM movies where title like 'Toy Story%';
2. SELECT \* FROM movies where director like 'John Lasseter';
3. SELECT \* FROM movies where director not like 'John Lasseter';
4. SELECT \* FROM movies where title like 'WALL%';

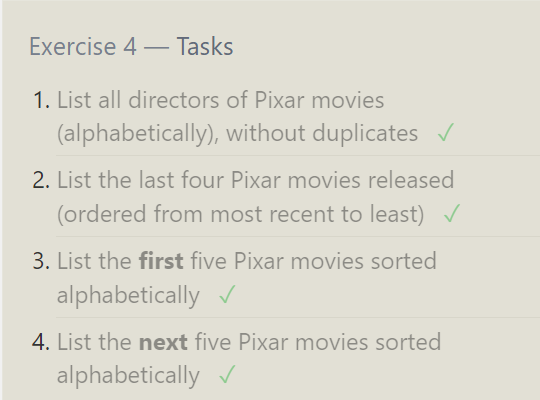
Result:



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

1. SELECT DISTINCT director FROM movies order by director Asc;
2. SELECT \* FROM movies order by year desc limit 4;
3. SELECT \* FROM movies order by title asc limit 5;
4. SELECT \* FROM movies order by title asc limit 5 offset 5;

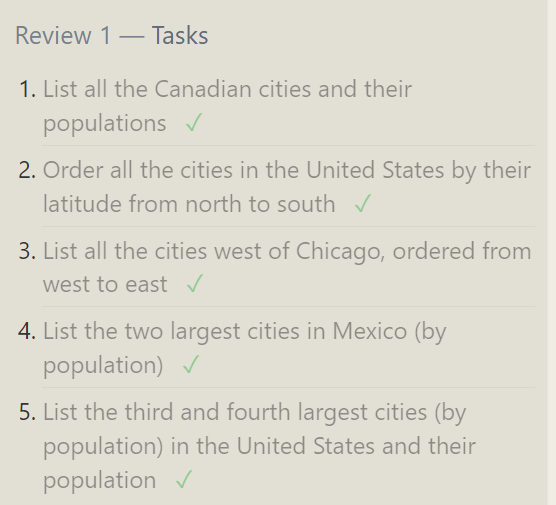
**Result:**

****

**SQL Review: Simple SELECT Queries**

1. SELECT City,population FROM north\_american\_cities where Country like 'Canada';
2. SELECT \* FROM north\_american\_cities where Country like 'United States' order by latitude desc
3. SELECT city FROM north\_american\_cities order by longitude asc limit 6
4. SELECT \* FROM north\_american\_cities where Country like "Mexico" order by population desc limit 2
5. SELECT \* FROM north\_american\_cities where Country like "United States" order by population desc limit 2 offset 2

Result:



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

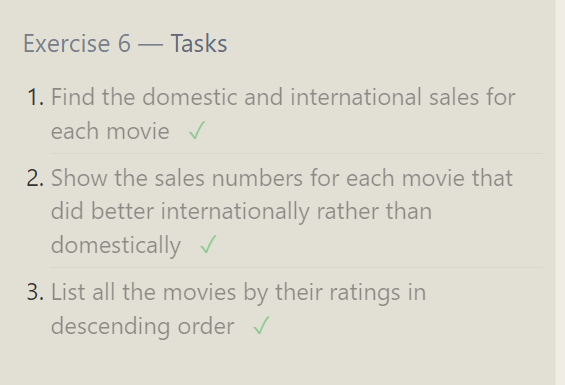
1. SELECT \* FROM movies inner join boxoffice on movies.id=boxoffice.movie\_id;
2. SELECT \* FROM movies inner join boxoffice on movies.id=boxoffice.movie\_id

where International\_sales>domestic\_Sales;

1. SELECT \* FROM movies inner join boxoffice on movies.id=boxoffice.movie\_id

order by rating desc;

**Result:**

****

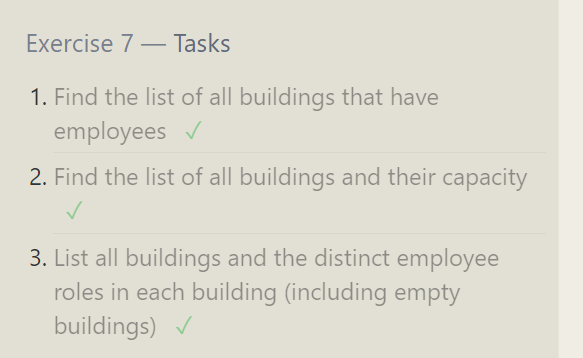
**SQL Lesson 7: OUTER JOINs**

1. SELECT DISTINCT Building\_name FROM buildings INNER JOIN Employees

on employees.building=buildings.building\_name;

1. SELECT \* FROM buildings;
2. SELECT distinct Role,building\_name FROM buildings LEFT JOIN employees on Employees.building=buildings.building\_name

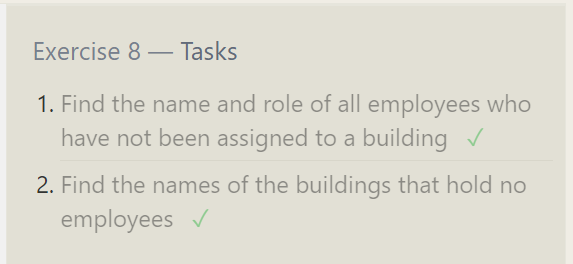
**Result:**

****

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

1. SELECT Name, Role FROM Employees WHERE Building IS NULL;
2. SELECT Building\_name FROM Buildings WHERE Building\_name NOT IN (SELECT DISTINCT Building FROM Employees WHERE Building IS NOT NULL);

Result:

****

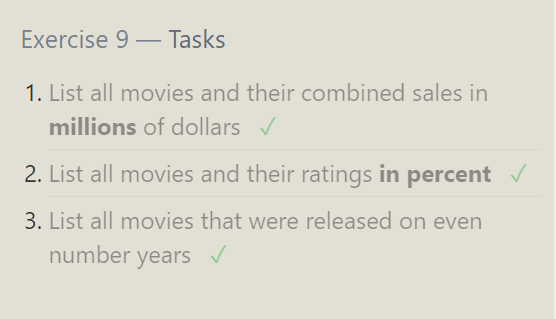
**SQL Lesson 9: Queries with expressions**

1. **SELECT m.Id,m.Title,m.Director, (b.Domestic\_sales + b.International\_sales) / 1000000 AS combined\_sales\_millions FROM Movies m JOIN Boxoffice b ON m.Id = b.Movie\_id;**

**2. SELECT m.Id, m.Title, m.Director, m.Year, m.Length\_minutes, b.Rating \* 10 AS rating\_percent FROM Movies m JOIN Boxoffice b ON m.Id = b.Movie\_id;**

1. **SELECT Id, Title, Director, Year, Length\_minutes FROM Movies WHERE Year % 2 = 0;**

**Result:**

****

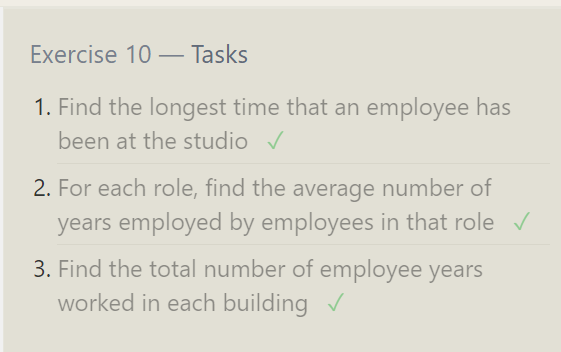
**SQL Lesson 10: Queries with aggregates (Pt. 1)**

1. SELECT MAX(Years\_employed) AS longest\_time FROM Employees;
2. SELECT Role, AVG(Years\_employed) AS avg\_years\_employed FROM Employees

GROUP BY Role;

1. SELECT Building, SUM(Years\_employed) AS total\_years\_worked FROM Employees GROUP BY Building;

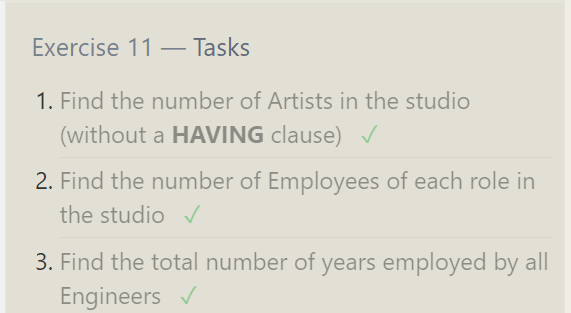
Result:



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

1. SELECT COUNT(\*) AS num\_artists FROM Employees WHERE Role = 'Artist';
2. SELECT Role, COUNT(\*) AS num\_employees FROM Employees GROUP BY Role;
3. SELECT SUM(Years\_employed) AS total\_years\_employed FROM Employees WHERE Role = 'Engineer';

Result:



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

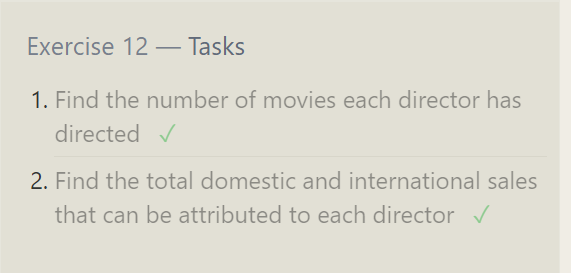
1. SELECT Director, COUNT(\*) AS NumMoviesDirected FROM Movies

GROUP BY Director;

1. SELECT m.Director, SUM(b.Domestic\_sales + b.International\_sales) AS TotalSales FROM Movies m JOIN Boxoffice b ON m.Id=b.Movie\_id

GROUP BY m.Director;

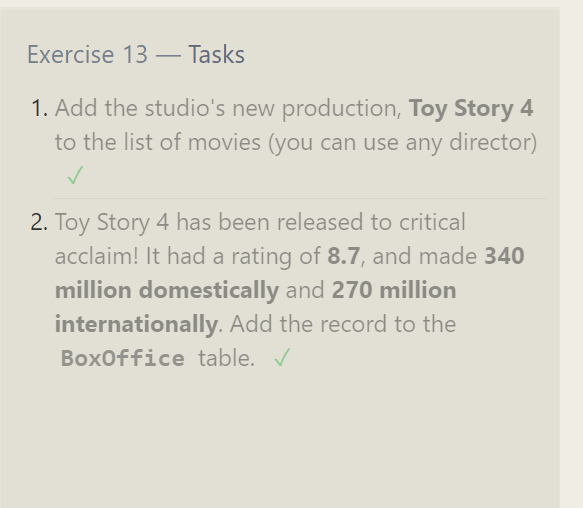
**Result:**

****

**SQL Lesson 13: Inserting rows**

1. Insert into movies values (4,'Toy Story 4','John Lasseter',2020,99)
2. insert into boxoffice values (4,8.7,34000000,27000000)

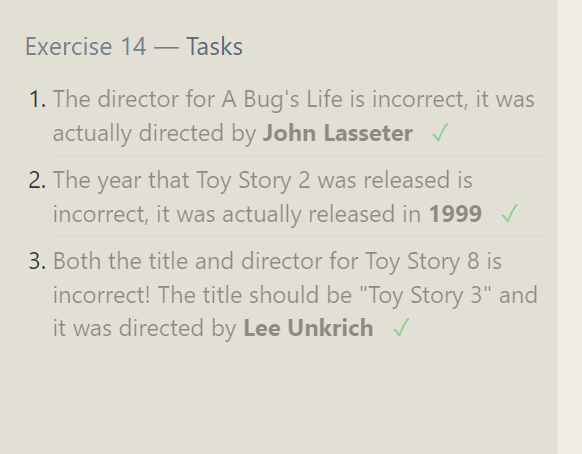
Result:



**SQL Lesson 14: Updating rows**

1. Update movies set Director='John Lasseter' where id=2
2. update movies set year='1999' where id=3
3. Update movies set title='Toy Story 3',Director='Lee Unkrich' where id=11

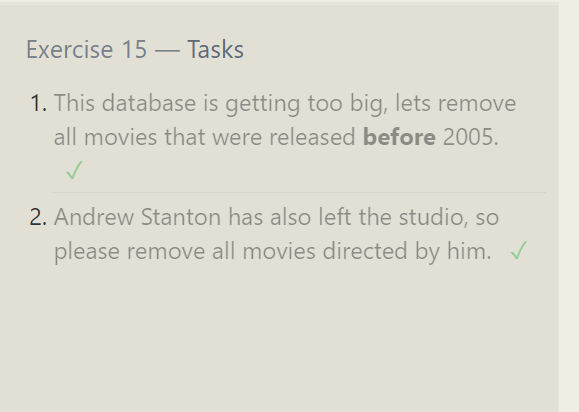
Result:



**SQL Lesson 15: Deleting rows**

1. delete from movies where year<2005
2. delete from movies where director='Andrew Stanton'

Result:



**SQL Lesson 16: Creating tables**

1. create table Database (

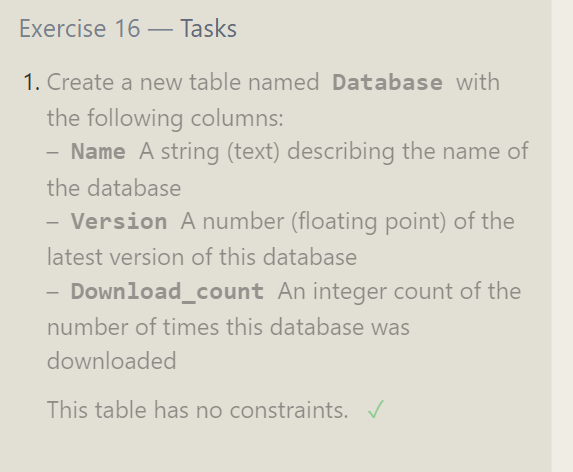
Name varchar(255),

Version float,

Download\_count int

);

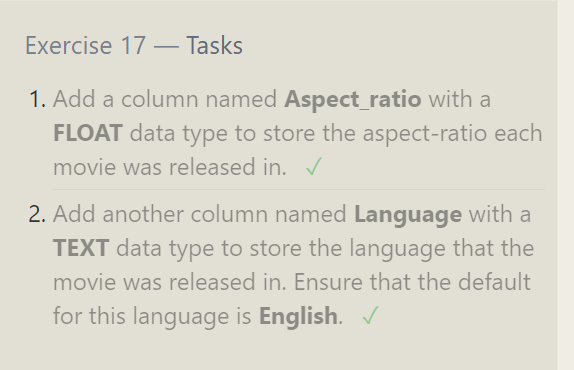
Result:



**SQL Lesson 17: Altering tables**

1. Alter table movies Add Aspect\_ratio float
2. Alter table movies Add Language varchar(255) not null default 'English'

Result:



**SQL Lesson 18: Dropping tables**

1. Drop table movies;
2. Drop table boxoffice;

Result:

