Exercise 1 — Tasks

1. Find the title of each film

```sql
SELECT Title FROM movies;
```

2. Find the director of each film

```sql
SELECT Director FROM movies;
```

3. Find the title and director of each film

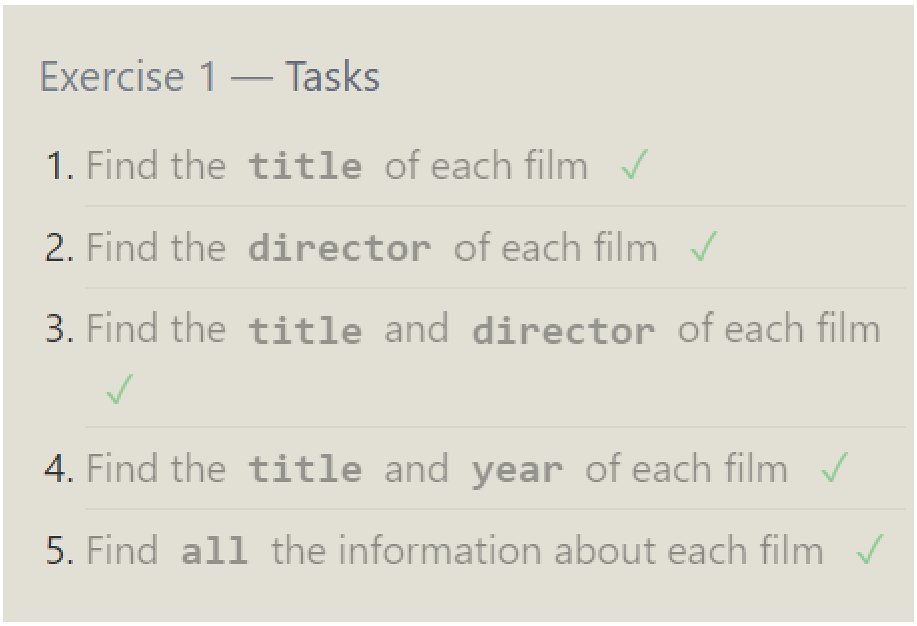
```sql
SELECT Title,Director FROM movies;
```

4. Find the title and year of each film

```sql
SELECT Title,Year FROM movies;
```

5. Find all the information about each film

```sql
SELECT * FROM movies;
```



Exercise 1 — Tasks

1. Find the `title` of each film ✓
2. Find the `director` of each film ✓
3. Find the `title` and `director` of each film ✓
4. Find the `title` and `year` of each film ✓
5. Find `all` the information about each film ✓

Exercise 2 — Tasks

1. Find the movie with a row id of 6

```
SELECT title FROM movies WHERE id=6;
```

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

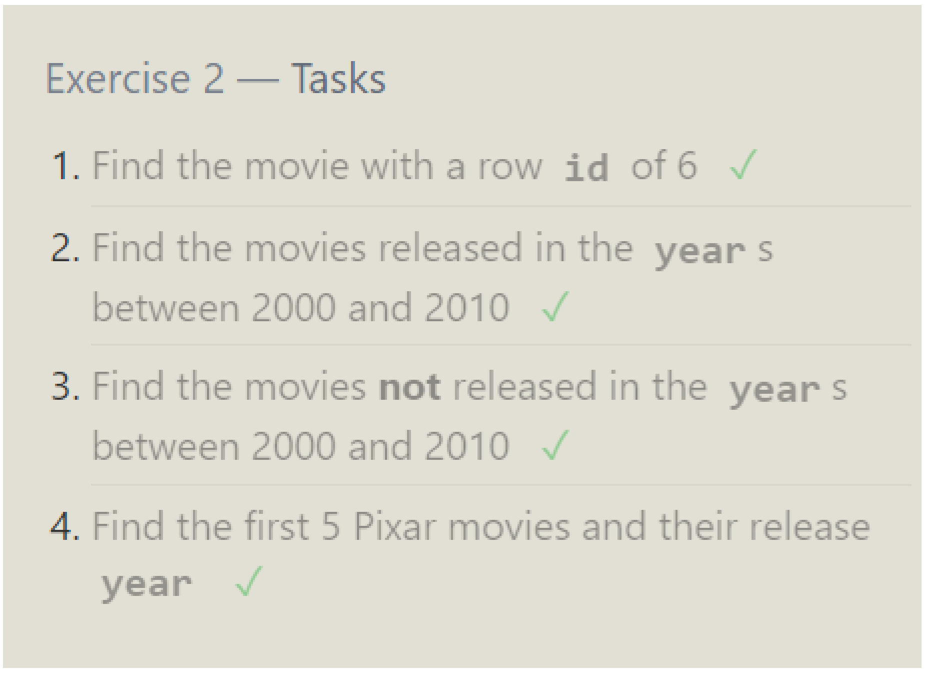
```
SELECT title FROM movies WHERE year BETWEEN 2000 AND 2010;
```

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

```
SELECT title FROM movies WHERE year NOT BETWEEN 2000 AND 2010;
```

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

```
SELECT title,year FROM movies LIMIT 5;
or
SELECT title,year FROM movies WHERE id BETWEEN 1 and 5;
```


Exercise 2 — Tasks

1. Find the movie with a row `id` of 6 ✓
2. Find the movies released in the `year`s between 2000 and 2010 ✓
3. Find the movies **not** released in the `year`s between 2000 and 2010 ✓
4. Find the first 5 Pixar movies and their release `year` ✓

Exercise 3 — Tasks

1. Find all the Toy Story movies

```
SELECT * FROM movies WHERE title LIKE "Toy Story%";
```

2. Find all the movies directed by John Lasseter

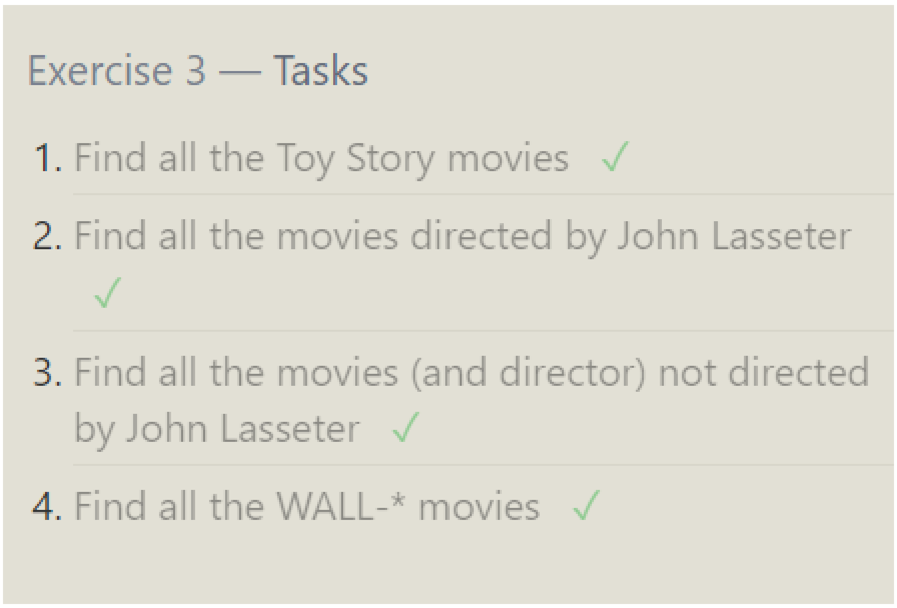
```
SELECT title FROM movies WHERE director = "John Lasseter";
```

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

```
SELECT title, director FROM movies WHERE director != "John Lasseter";
```

4. Find all the WALL-* movies

```
SELECT title FROM movies WHERE title LIKE "WALL-%";
```


Exercise 3 — Tasks

1. Find all the Toy Story movies ✓
2. Find all the movies directed by John Lasseter ✓
3. Find all the movies (and director) not directed by John Lasseter ✓
4. Find all the WALL-* movies ✓

Note : Like is case insensitive, equalto is case sensitive % - 0 or more, _ is exactly 1

Exercise 4 — Tasks

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

```
SELECT DISTINCT director FROM movies ORDER BY director; -- By default in ascending
order
```

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

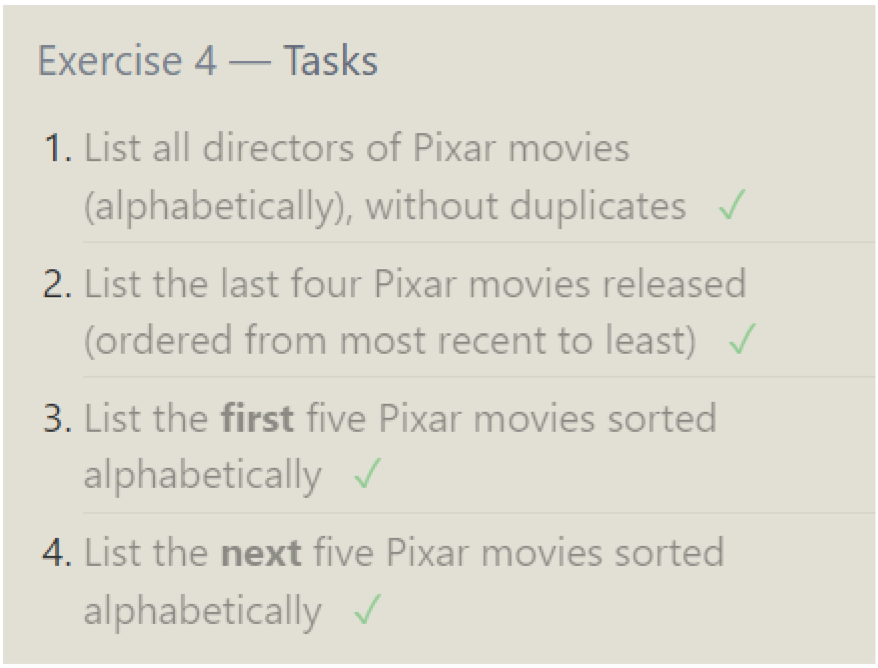
```
SELECT title FROM movies ORDER BY year DESC LIMIT 4;
```

3. List the first five Pixar movies sorted alphabetically

```
SELECT title FROM movies ORDER BY title LIMIT 5;
```

4. List the next five Pixar movies sorted alphabetically

```
SELECT title FROM movies ORDER BY title LIMIt 5 OFFSET 5;
```


Exercise 4 — Tasks

1. List all directors of Pixar movies (alphabetically), without duplicates ✓
2. List the last four Pixar movies released (ordered from most recent to least) ✓
3. List the **first** five Pixar movies sorted alphabetically ✓
4. List the **next** five Pixar movies sorted alphabetically ✓

Review 1 — Tasks

1. List all the Canadian cities and their populations

```
SELECT city,population FROM north_american_cities WHERE country="Canada";
```

2. 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;
```

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

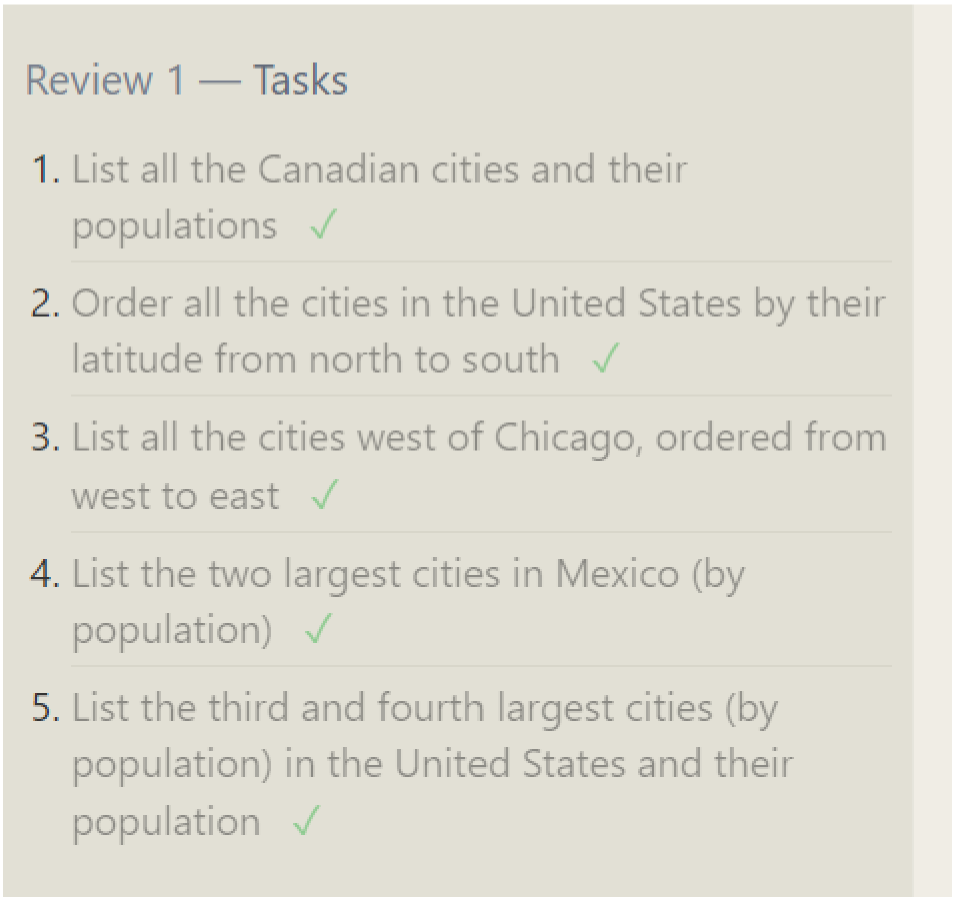
```
SELECT city FROM north_american_cities ORDER BY longitude limit 6;

-- SELECT * FROM north_american_cities WHERE longitude < (select longitude from
north_american_cities where city = "Chicago") order by longitude;
```

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

```
SELECT city 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 city FROM north_american_cities WHERE country="United States" ORDER BY
population desc LIMIT 2 OFFSET 2;
```



Review 1 — Tasks

1. List all the Canadian cities and their populations ✓
2. Order all the cities in the United States by their latitude from north to south ✓
3. List all the cities west of Chicago, ordered from west to east ✓
4. List the two largest cities in Mexico (by population) ✓
5. List the third and fourth largest cities (by population) in the United States and their population ✓

Exercise 6 — Tasks

> Why we use only INNER JOIN?
> Because Left join is not possible as there will be no data of sales without the movies got released. And Right Joinn is not needed as there will be no sales data when there are no corresponding movie.

1. Find the domestic and international sales for each movie

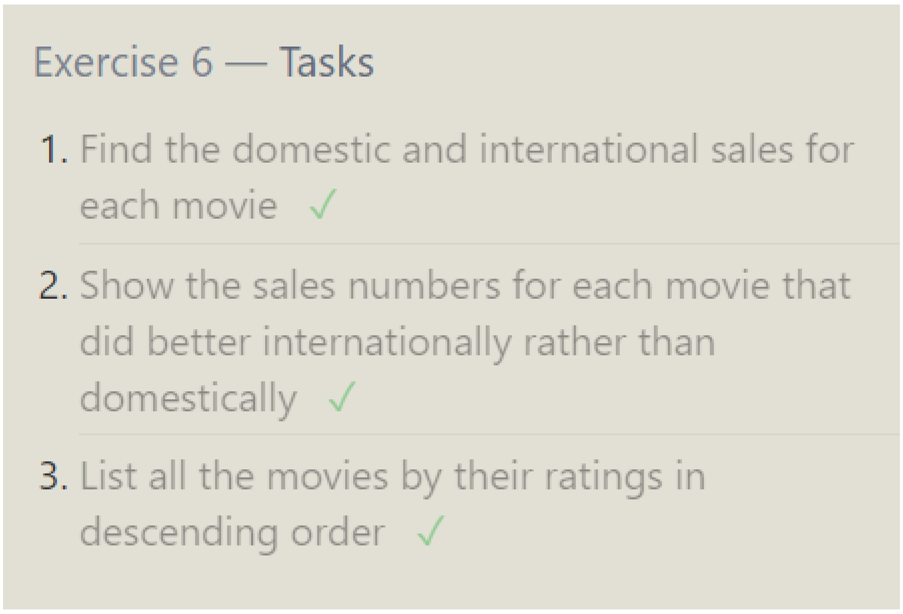
```
SELECT title, domestic_sales, International_sales
FROM Boxoffice as bo
INNER JOIN movies as mv
ON bo.Movie_id = mv.id;
```

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

```
SELECT title, domestic_sales, International_sales
FROM Boxoffice as bo
INNER JOIN movies as mv
ON bo.Movie_id = mv.id
WHERE international_sales > domestic_sales;
```

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

```
SELECT title, domestic_sales, International_sales
FROM Boxoffice as bo
INNER JOIN movies as mv
ON bo.Movie_id = mv.id
ORDER BY rating DESC;
```


Exercise 6 — Tasks

1. Find the domestic and international sales for each movie ✓
2. Show the sales numbers for each movie that did better internationally rather than domestically ✓
3. List all the movies by their ratings in descending order ✓

Exercise 7 — Tasks

1. Find the list of all buildings that have employees

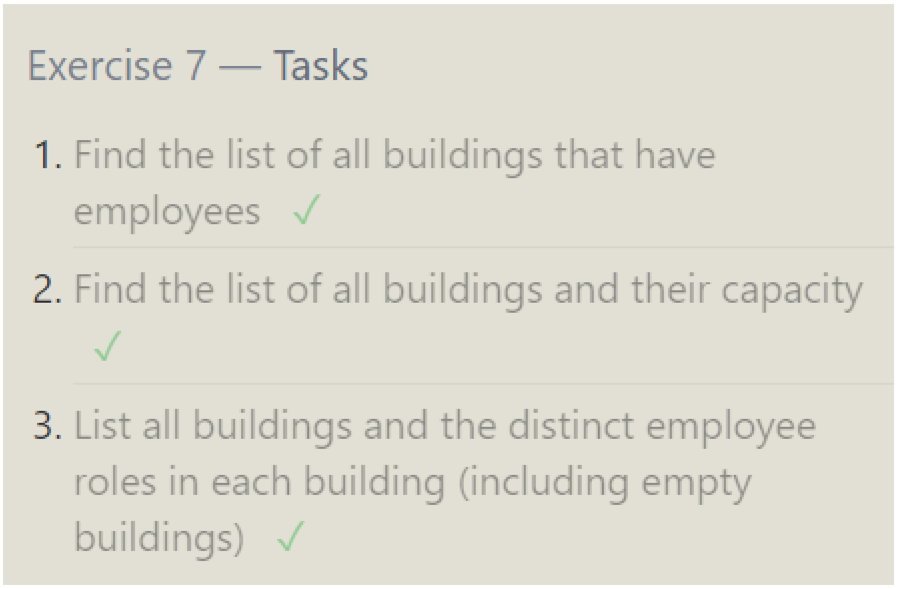
```
SELECT DISTINCT building
FROM employees;
```

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)

```sql
SELECT Distinct building_name, role
FROM buildings as bd
LEFT JOIN Employees as emp
ON bd.building_name = emp.building;
```



Exercise 7 — Tasks

1. Find the list of all buildings that have employees ✓
2. Find the list of all buildings and their capacity ✓
3. List all buildings and the distinct employee roles in each building (including empty buildings) ✓

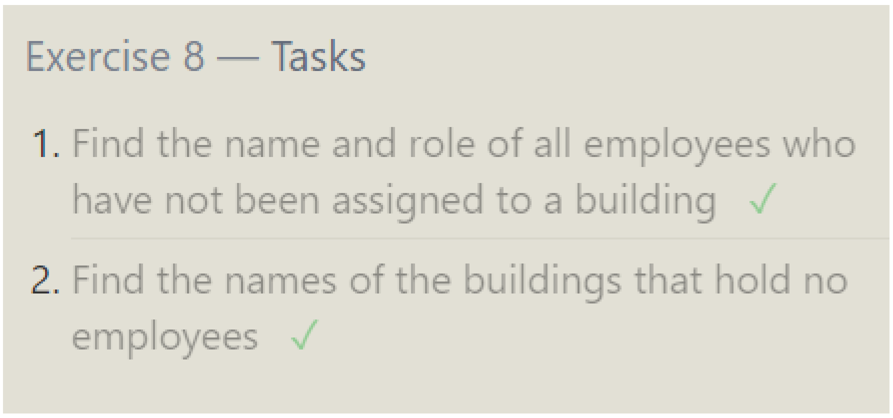
Exercise 8 — Tasks

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

```sql
SELECT name,role
FROM employees
WHERE building IS NULL;
```

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

```sql
SELECT Distinct building_name, role
FROM buildings as bd
LEFT JOIN Employees as emp
ON bd.building_name = emp.building WHERE role IS NULL;
```



Exercise 8 — Tasks

1. Find the name and role of all employees who have not been assigned to a building ✓
2. Find the names of the buildings that hold no employees ✓

Exercise 9 — Tasks

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

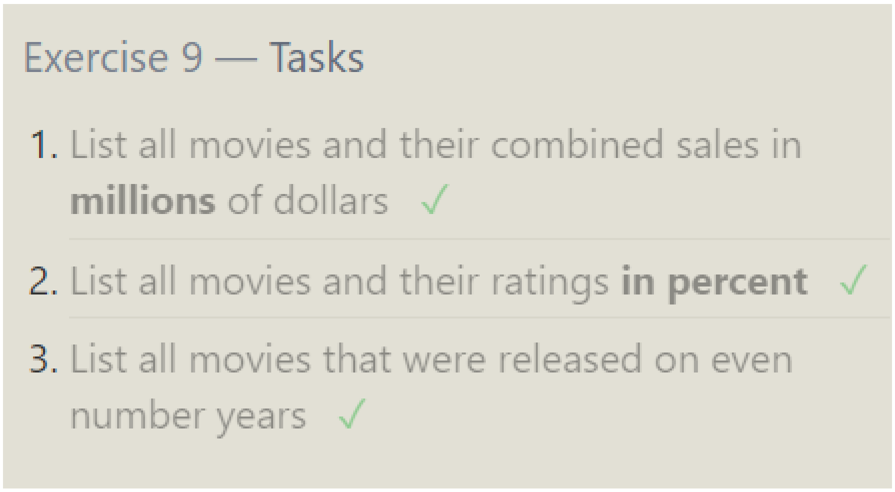
```sql
SELECT Title, (Domestic_sales+International_sales)/1000000 as Combined_Sales
FROM movies as mv
INNER JOIN Boxoffice as bo
ON mv.Id = bo.Movie_id;
```

2. List all movies and their ratings in percent

```sql
SELECT Title, rating*10 as Ratings_in_Percentage
FROM movies as mv
INNER JOIN Boxoffice as bo
ON mv.Id = bo.Movie_id;
```

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

```sql
SELECT Title FROM movies WHERE Year%2==0;
```



Exercise 9 — Tasks

1. List all movies and their combined sales in **millions** of dollars ✓
2. List all movies and their ratings **in percent** ✓
3. List all movies that were released on even number years ✓

Exercise 10 — Tasks

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

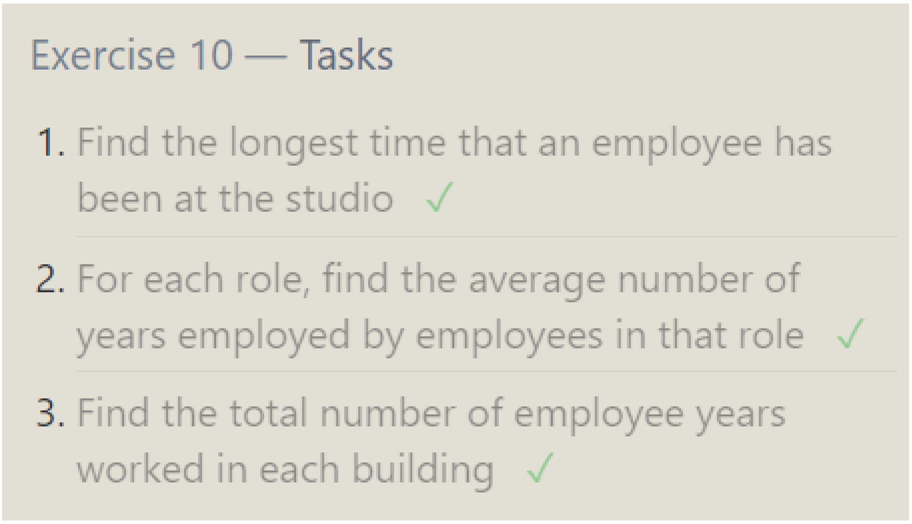
```
SELECT role, max(years_employed) as Longest_Time_in_Studio FROM employees;
```

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

```
SELECT role,AVG(Years_employed)
FROM employees
WHERE Years_employed
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;
```


Exercise 10 — Tasks

1. Find the longest time that an employee has been at the studio ✓
2. For each role, find the average number of years employed by employees in that role ✓
3. Find the total number of employee years worked in each building ✓

Exercise 11 — Tasks

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

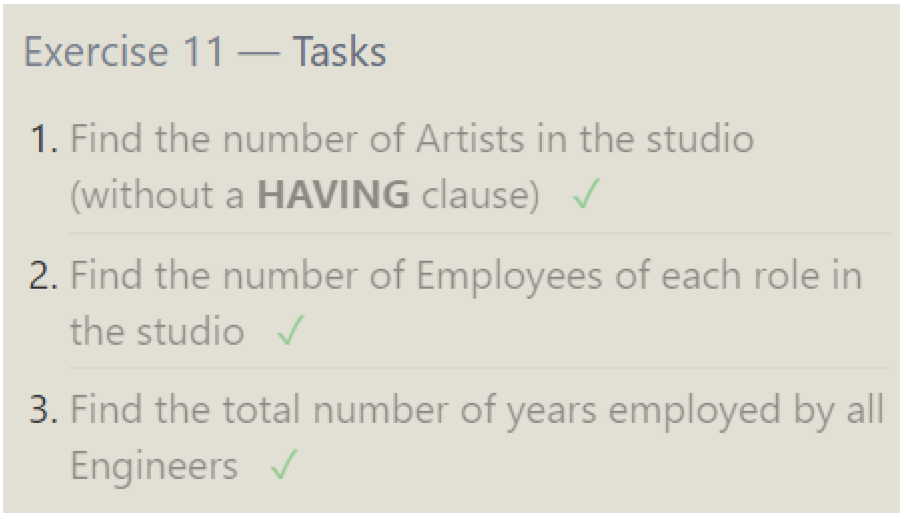
```
2. SELECT count(role) FROM employees WHERE role LIKE 'A%';
```

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

```
SELECT role, count(name) FROM employees GROUP BY role;
```

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

```
SELECT role,sum(Years_employed)
FROM employees
WHERE role="Engineer";
```



Exercise 11 — Tasks

1. Find the number of Artists in the studio (without a **HAVING** clause) ✓
2. Find the number of Employees of each role in the studio ✓
3. Find the total number of years employed by all Engineers ✓

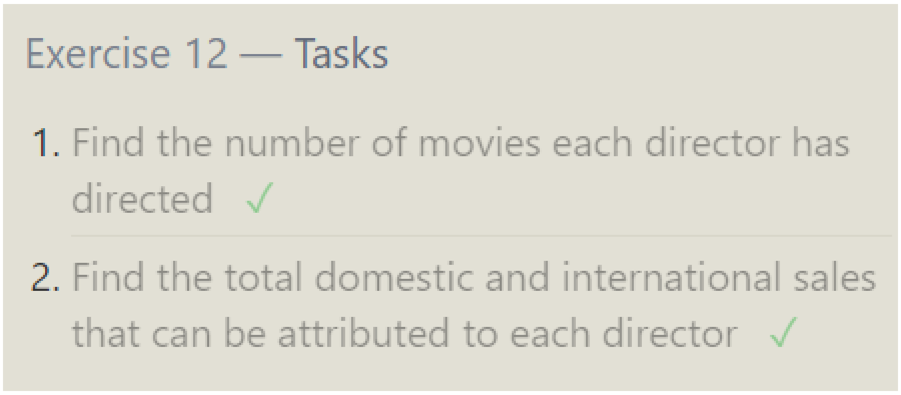
Exercise 12 — Tasks

1. Find the number of movies each director has directed

```
SELECT Director, count(title) As Movies_Directed FROM movies GROUP BY Director;
```

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

```
SELECT Director, Domestic_sales, International_sales,
SUM(Domestic_sales+International_sales) as Total_Sales
FROM movies as mv
JOIN Boxoffice as bo
ON id = Movie_id
GROUP BY Director;
```



Exercise 12 — Tasks

1. Find the number of movies each director has directed ✓
2. Find the total domestic and international sales that can be attributed to each director ✓

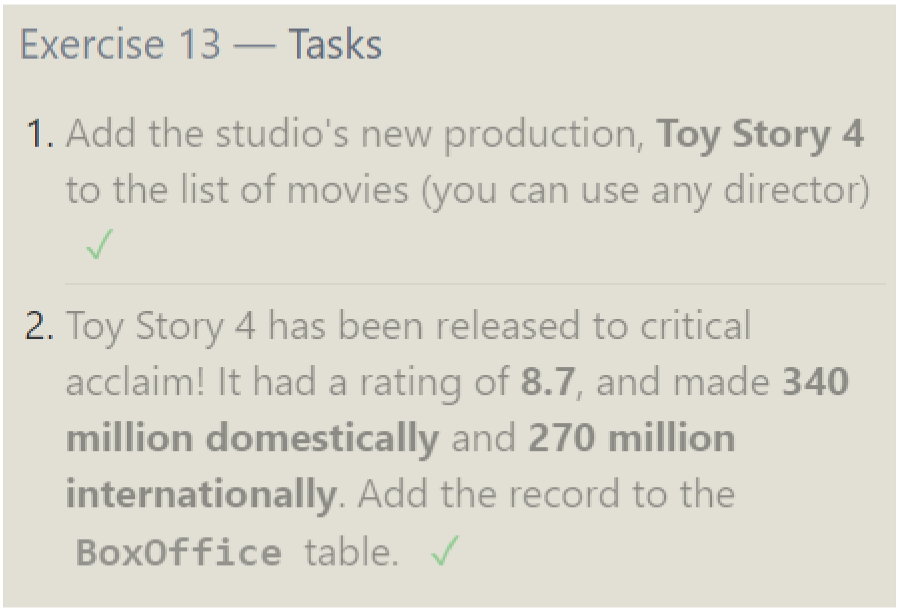
Exercise 13 — Tasks

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

```sql
INSERT INTO Movies (Id, Title, Director, Year, Length_minutes)
VALUES (15, "Toy Story 4", "John Lasseter", 2015, 105);
```

2. 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
INSERT INTO Boxoffice VALUES (15, 8.7, 340000000, 270000000);
```


Exercise 13 — Tasks

1. Add the studio's new production, **Toy Story 4** to the list of movies (you can use any director) ✓
2. 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. ✓

Exercise 14 — Tasks

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

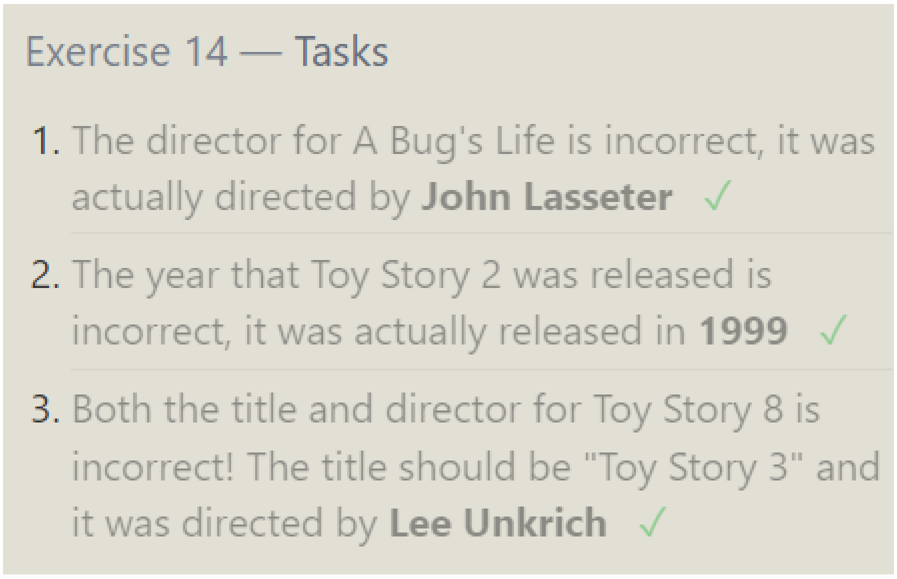
```sql
UPDATE Movies
SET Director = "John Lasseter"
WHERE Title = "A Bug's Life";
```

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

```sql
UPDATE Movies
SET Year = 1999
WHERE Title = "Toy Story 2";
```

3. 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
UPDATE Movies
SET Title = "Toy Story 3", Director = "Lee Unkrich"
WHERE Title = "Toy Story 8";
```



Exercise 14 — Tasks

1. The director for A Bug's Life is incorrect, it was actually directed by **John Lasseter** ✓
2. The year that Toy Story 2 was released is incorrect, it was actually released in **1999** ✓
3. 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** ✓

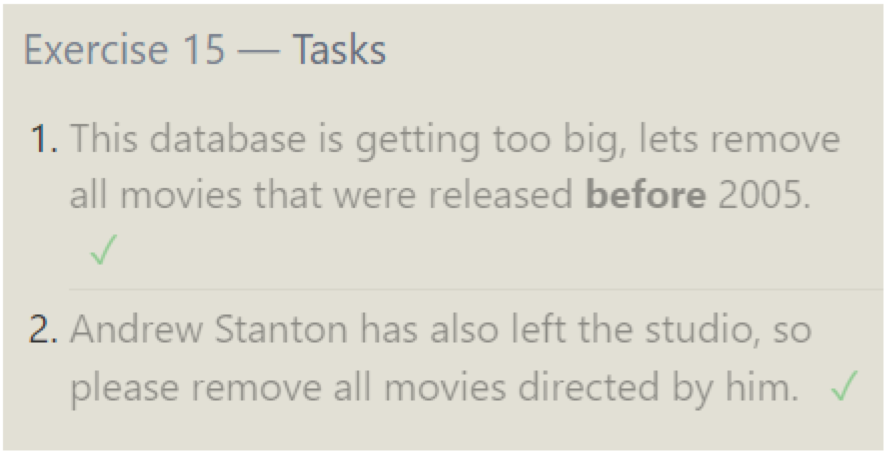
Exercise 15 — Tasks

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

```sql
DELETE FROM movies
WHERE Year<2005;
```

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

```sql
DELETE FROM movies
WHERE Director = "Andrew Stanton";
```



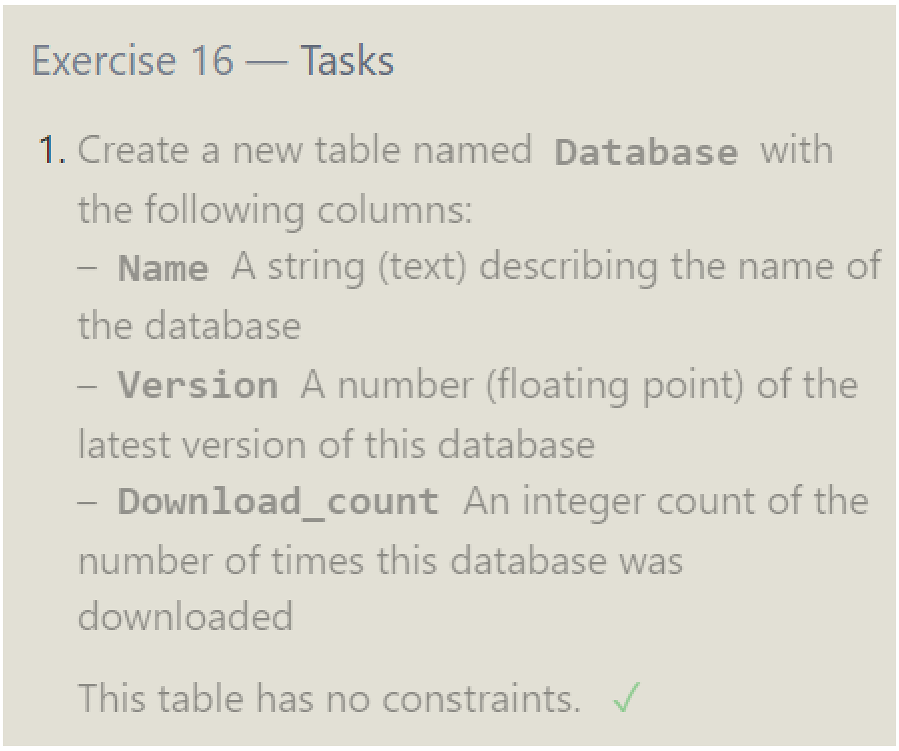
Exercise 15 — Tasks

1. This database is getting too big, lets remove all movies that were released **before** 2005. ✓
2. Andrew Stanton has also left the studio, so please remove all movies directed by him. ✓

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 VARCHAR,
                        Version FLOAT,
                        Download_count int);
```



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. ✓

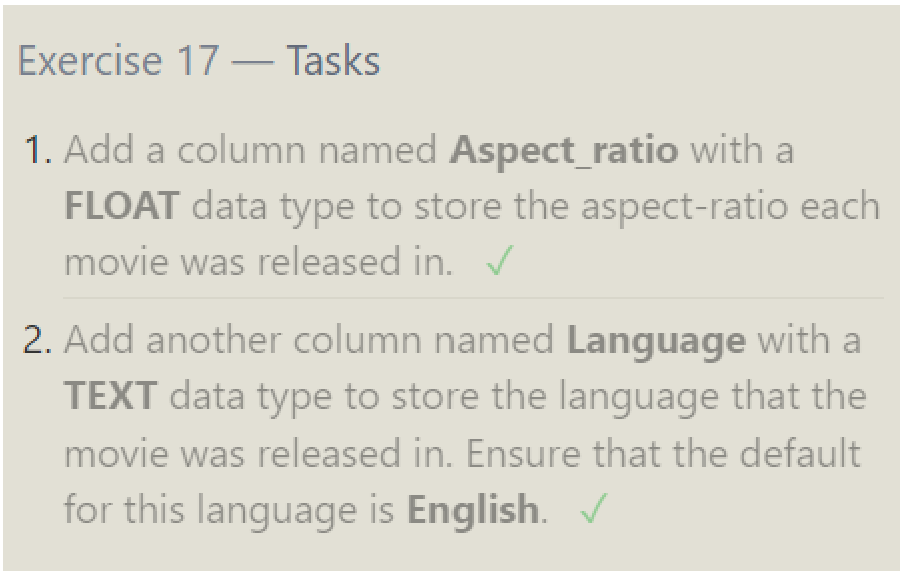
Exercise 17 — Tasks

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 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
DEFAULT English;
```



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. ✓
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**. ✓

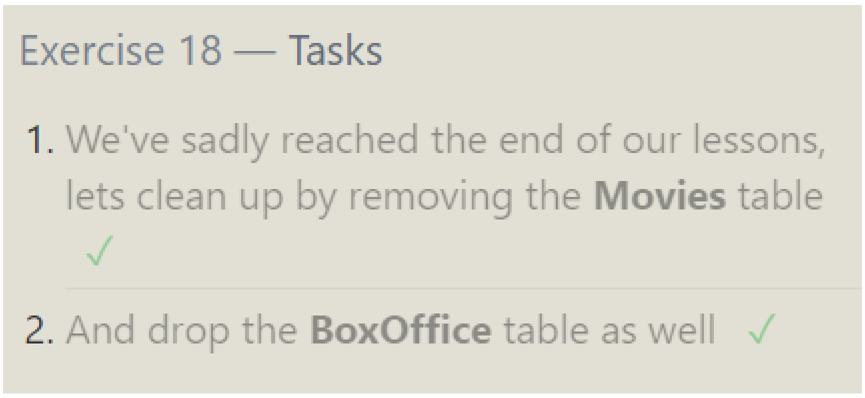
Exercise 18 — Tasks

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

```sql
DROP TABLE movies;
```

2. And drop the BoxOffice table as well

```sql
DROP TABLE BoxOffice;
```



Exercise 18 — Tasks

1. We've sadly reached the end of our lessons, lets clean up by removing the **Movies** table ✓
2. And drop the **BoxOffice** table as well ✓