SQL Bolt Exercises

By Karthikeyan Meenakshi Sundaram

SQL Bolt website

SQLBolt - Learn SQL - Introduction to SQL

Welcome to SQLBolt, a series of interactive lessons and exercises designed to help you quickly learn SQL right in your browser. SQL, or Structured Query Language, is a language designed to allow both technical and non-technical users query, manipulate, and transform data from a relational database.

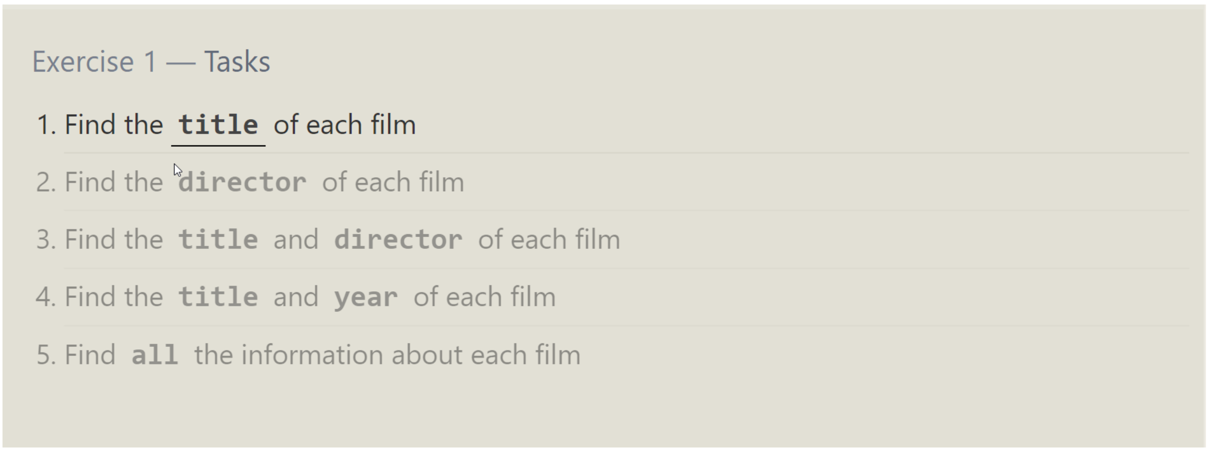
https://sqlbolt.com/lesson/introduction

Exercise 1

SQLBolt - Learn SQL - SQL Lesson 1: SELECT queries 101

To retrieve data from a SQL database, we need to write SELECT statements, which are often colloquially refered to as queries. A query in itself is just a statement which declares what data we are looking for, where to find it in the database, and optionally, how to transform it before it is returned.

https://sqlbolt.com/lesson/select_queries_introduction



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

1(i).Find the `title` of each film

```
SELECT title FROM movies;
```

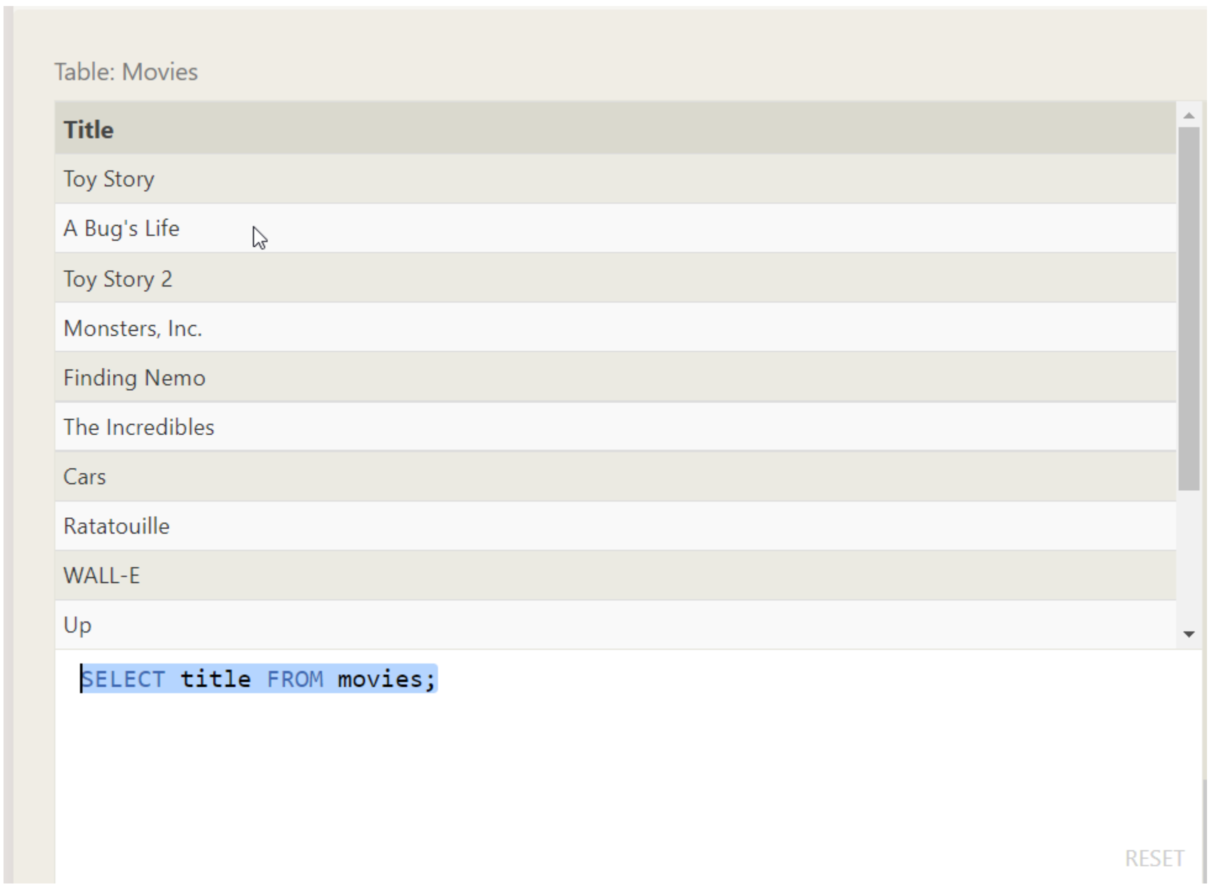


Table: Movies

Title
Toy Story
A Bug's Life
Toy Story 2
Monsters, Inc.
Finding Nemo
The Incredibles
Cars
Ratatouille
WALL-E
Up

```
SELECT title FROM movies;
```

1(ii). Find the `director` of each film

```
SELECT director FROM movies;
```

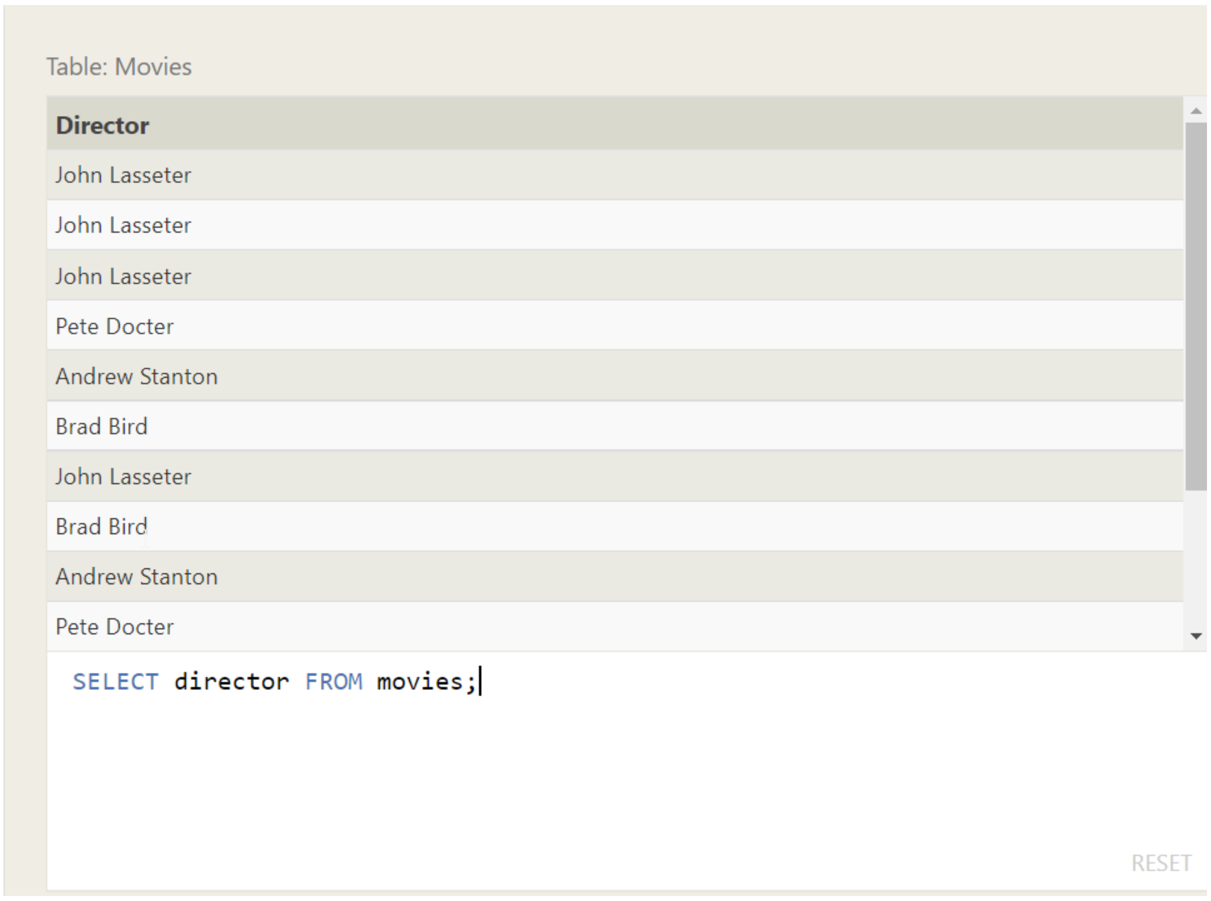


Table: Movies

Director
John Lasseter
John Lasseter
John Lasseter
Pete Docter
Andrew Stanton
Brad Bird
John Lasseter
Brad Bird
Andrew Stanton
Pete Docter

```
SELECT director FROM movies;
```

1(iii). Find the `title` and `director` of each film

```
SELECT title,director FROM movies;
```

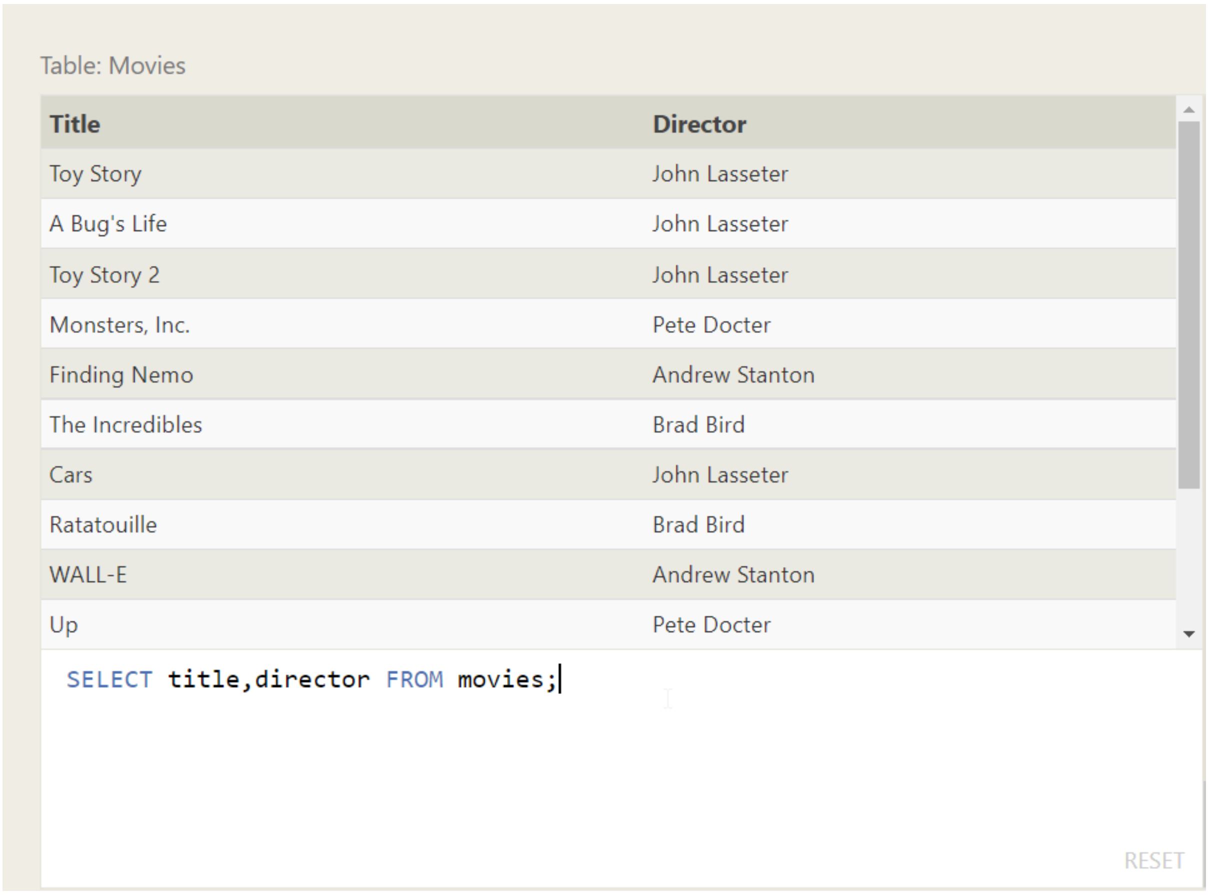


Table: Movies

Title	Director
Toy Story	John Lasseter
A Bug's Life	John Lasseter
Toy Story 2	John Lasseter
Monsters, Inc.	Pete Docter
Finding Nemo	Andrew Stanton
The Incredibles	Brad Bird
Cars	John Lasseter
Ratatouille	Brad Bird
WALL-E	Andrew Stanton
Up	Pete Docter

```
SELECT title,director FROM movies;
```

1(iv).Find the `title` and `year` of each film

```
SELECT title,year FROM movies;
```

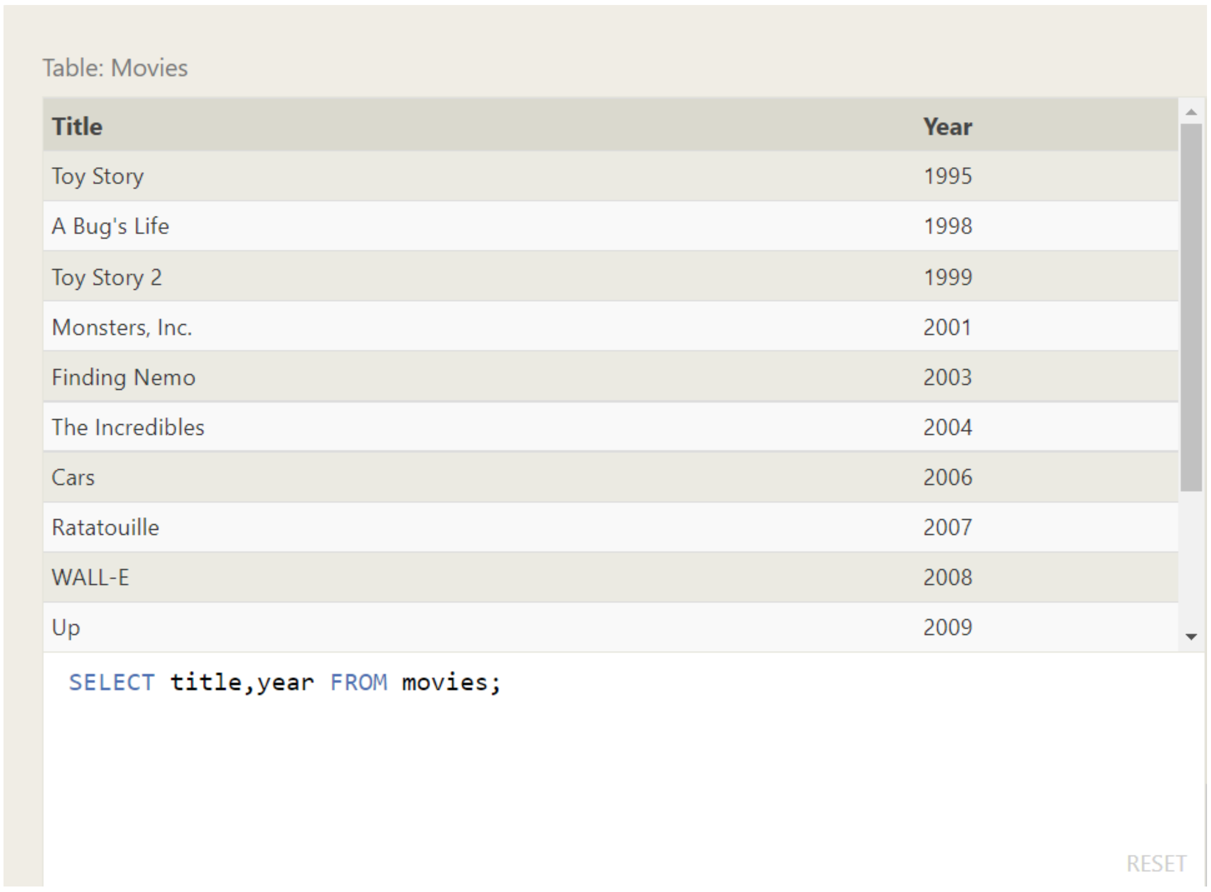


Table: Movies

Title	Year
Toy Story	1995
A Bug's Life	1998
Toy Story 2	1999
Monsters, Inc.	2001
Finding Nemo	2003
The Incredibles	2004
Cars	2006
Ratatouille	2007
WALL-E	2008
Up	2009

```
SELECT title,year FROM movies;
```

1(v).Find `all` the information about each film

```
SELECT * FROM movies;
```

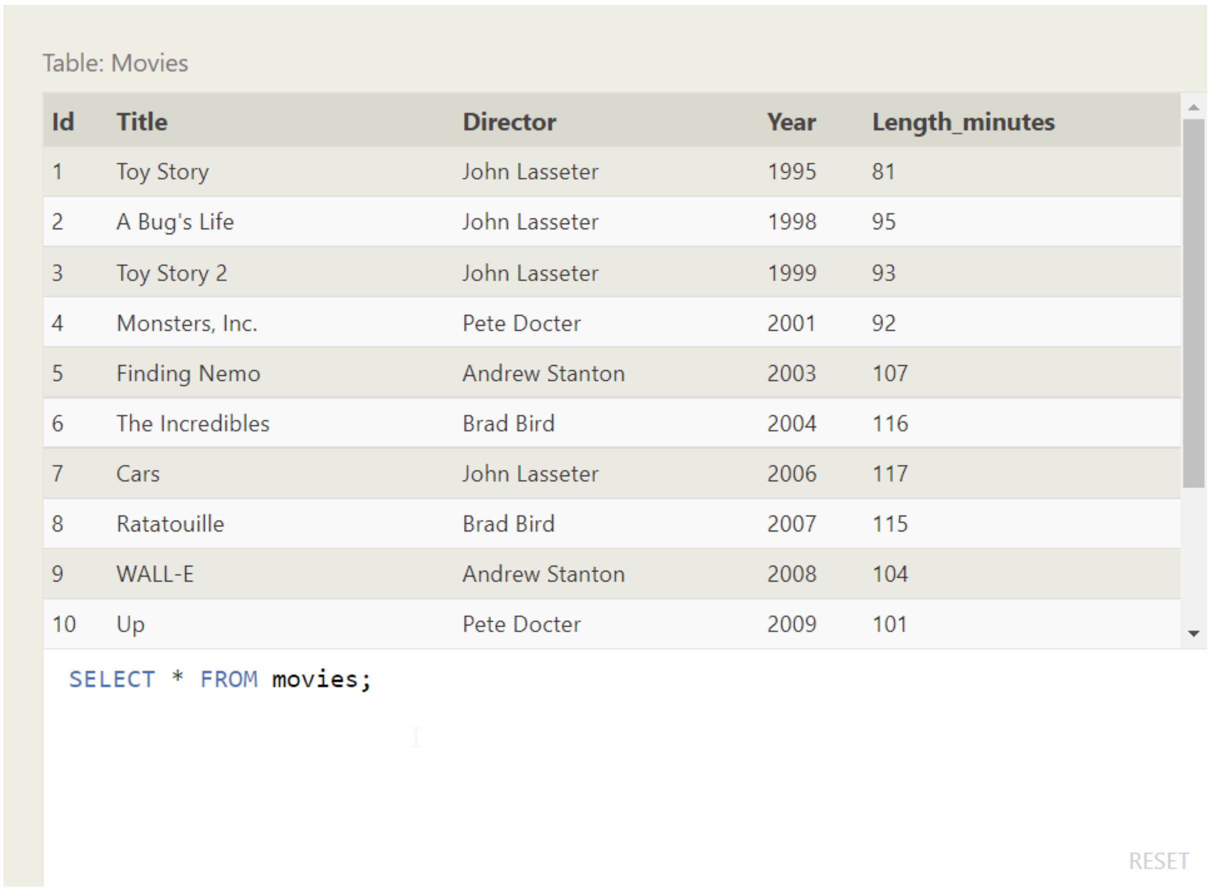


Table: Movies

Id	Title	Director	Year	Length_minutes
1	Toy Story	John Lasseter	1995	81
2	A Bug's Life	John Lasseter	1998	95
3	Toy Story 2	John Lasseter	1999	93
4	Monsters, Inc.	Pete Docter	2001	92
5	Finding Nemo	Andrew Stanton	2003	107
6	The Incredibles	Brad Bird	2004	116
7	Cars	John Lasseter	2006	117
8	Ratatouille	Brad Bird	2007	115
9	WALL-E	Andrew Stanton	2008	104
10	Up	Pete Docter	2009	101

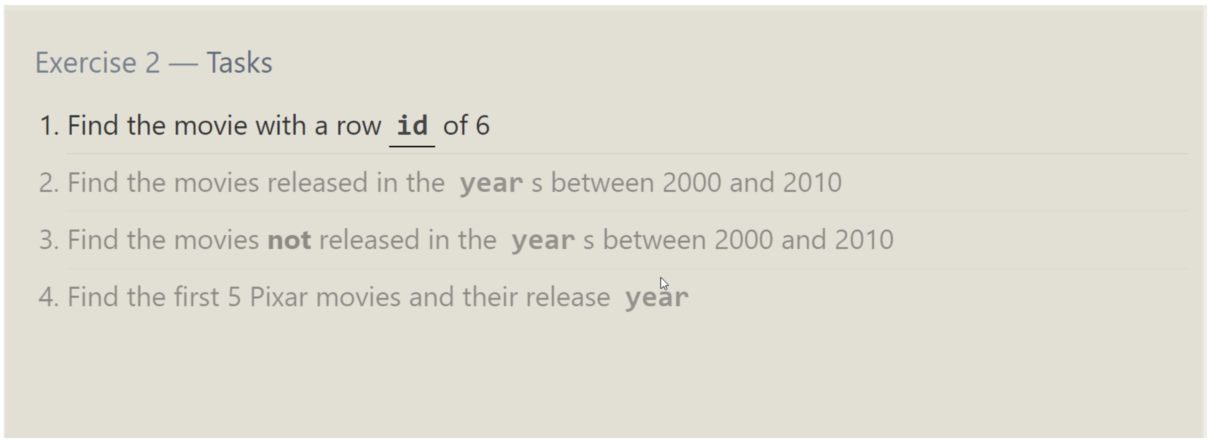
```
SELECT * FROM movies;
```

Exercise 2

SQLBolt - Learn SQL - SQL Lesson 2: Queries with constraints (Pt. 1)

Now we know how to select for specific columns of data from a table, but if you had a table with a hundred million rows of data, reading through all the rows would be inefficient and perhaps even impossible. In order to filter certain results from being returned, we need to use a WHERE clause in the query.

https://sqlbolt.com/lesson/select_queries_with_constraints



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`

2(i). Find the movie with a row `id` of 6

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

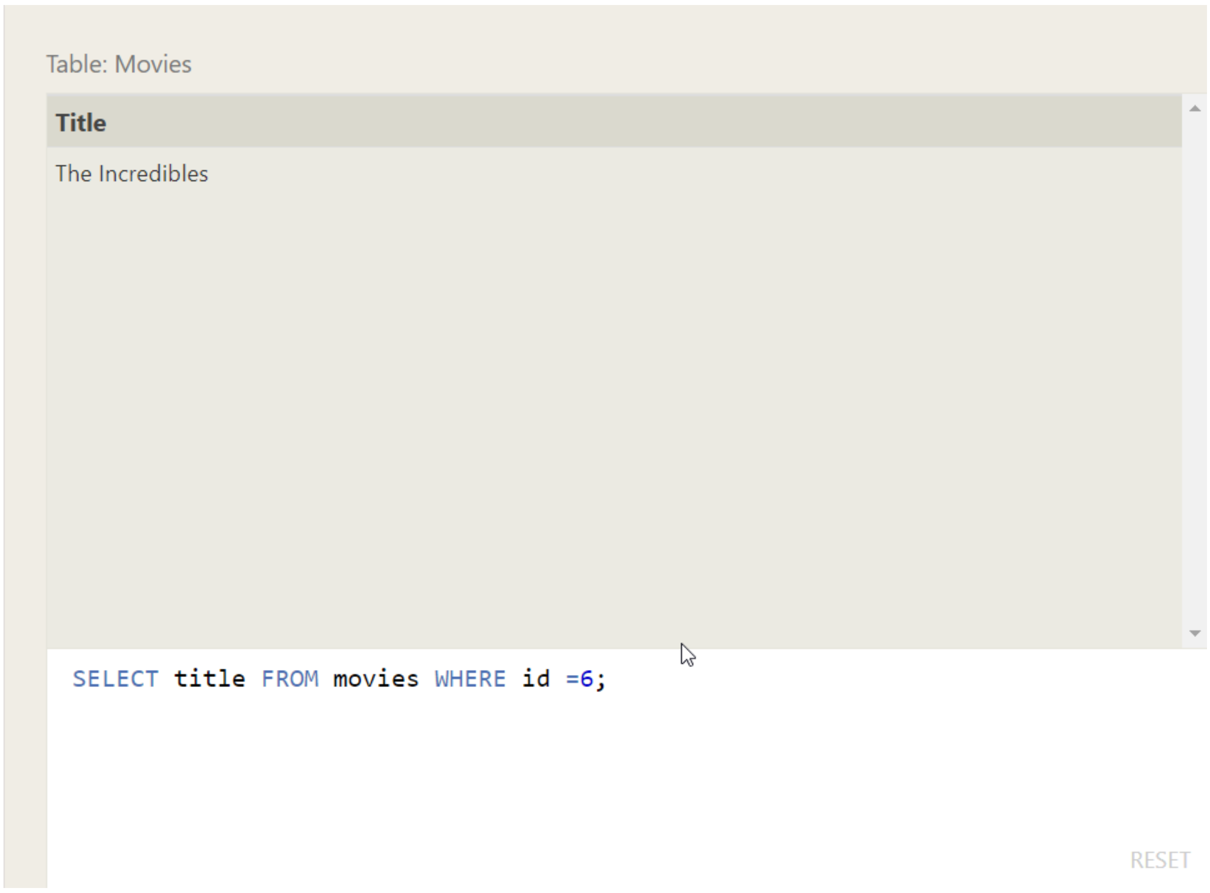


Table: Movies

Title
The Incredibles

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

2(ii). Find the movies released in the `years` between 2000 and 2010

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

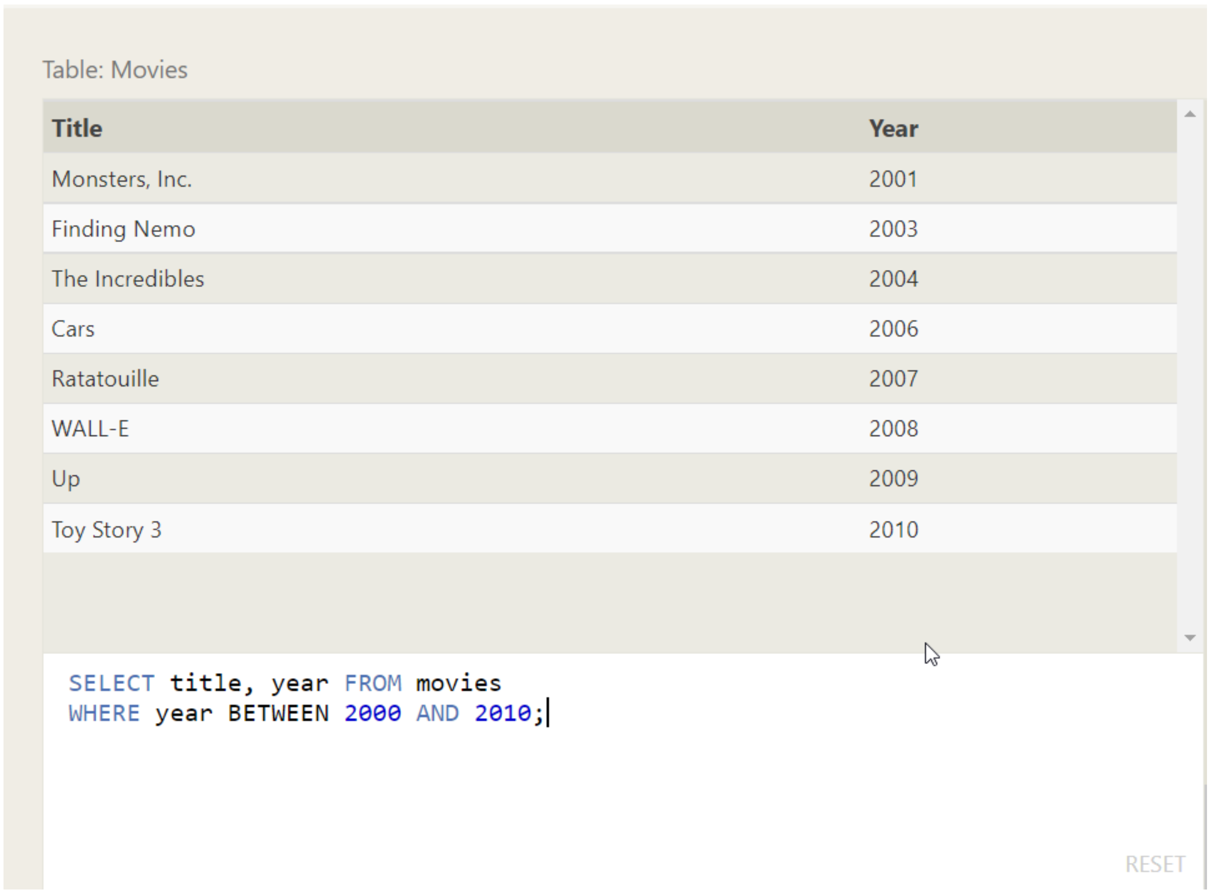


Table: Movies

Title	Year
Monsters, Inc.	2001
Finding Nemo	2003
The Incredibles	2004
Cars	2006
Ratatouille	2007
WALL-E	2008
Up	2009
Toy Story 3	2010

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

2(iii). Find the movies not released in the `year`s between 2000 and 2010

```
SELECT title, year FROM movies
WHERE year < 2000 OR year > 2010;
```

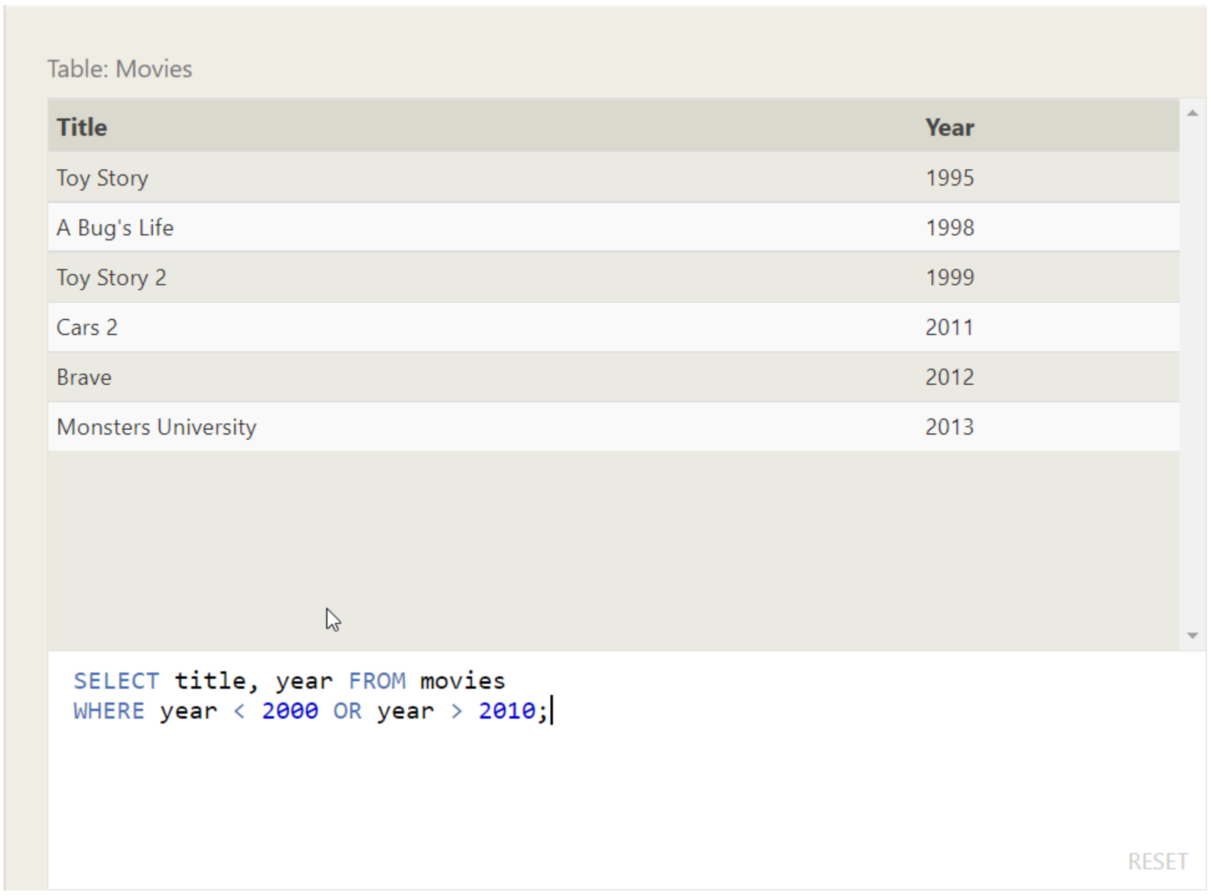


Table: Movies

Title	Year
Toy Story	1995
A Bug's Life	1998
Toy Story 2	1999
Cars 2	2011
Brave	2012
Monsters University	2013

```
SELECT title, year FROM movies
WHERE year < 2000 OR year > 2010;
```

2(iv).Find the first 5 Pixar movies and their release `year`

```
SELECT title,year FROM movies WHERE id<6
```

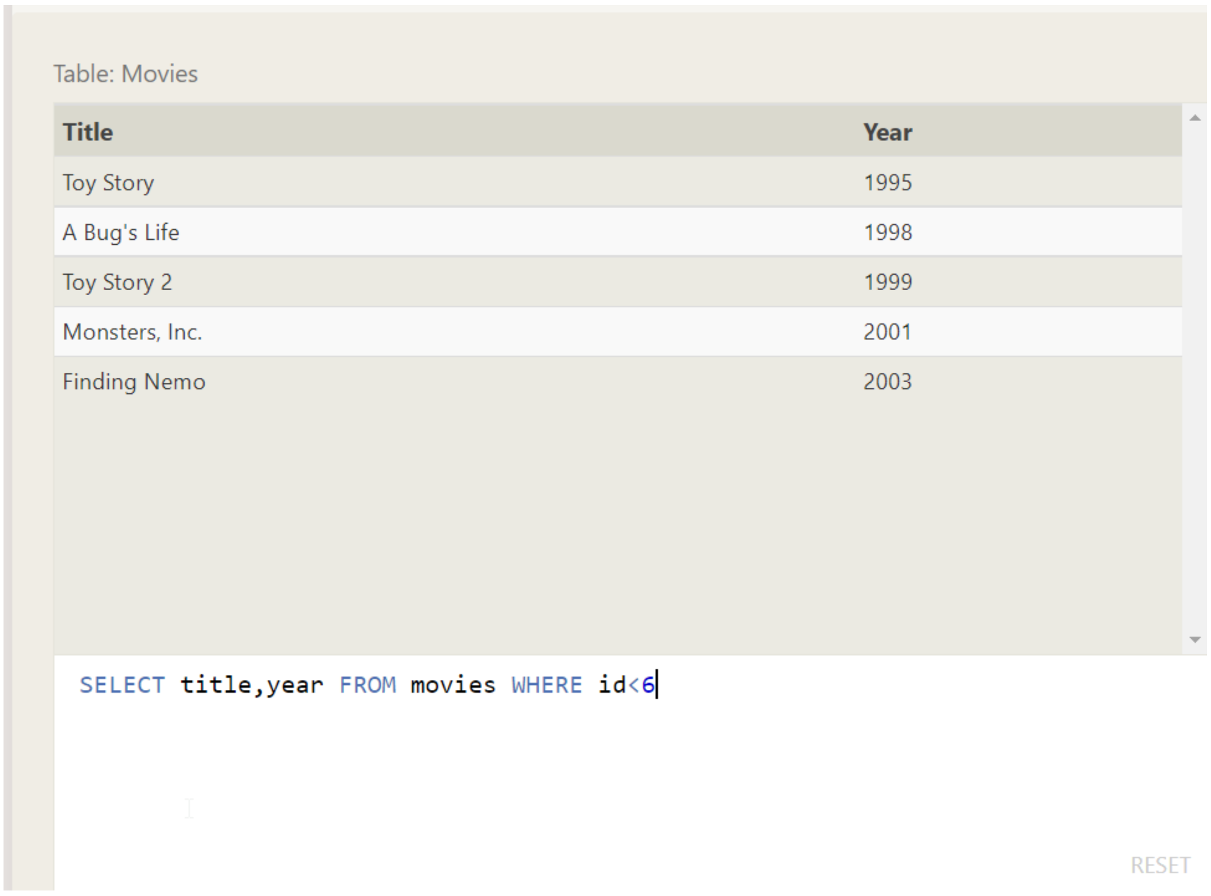


Table: Movies

Title	Year
Toy Story	1995
A Bug's Life	1998
Toy Story 2	1999
Monsters, Inc.	2001
Finding Nemo	2003

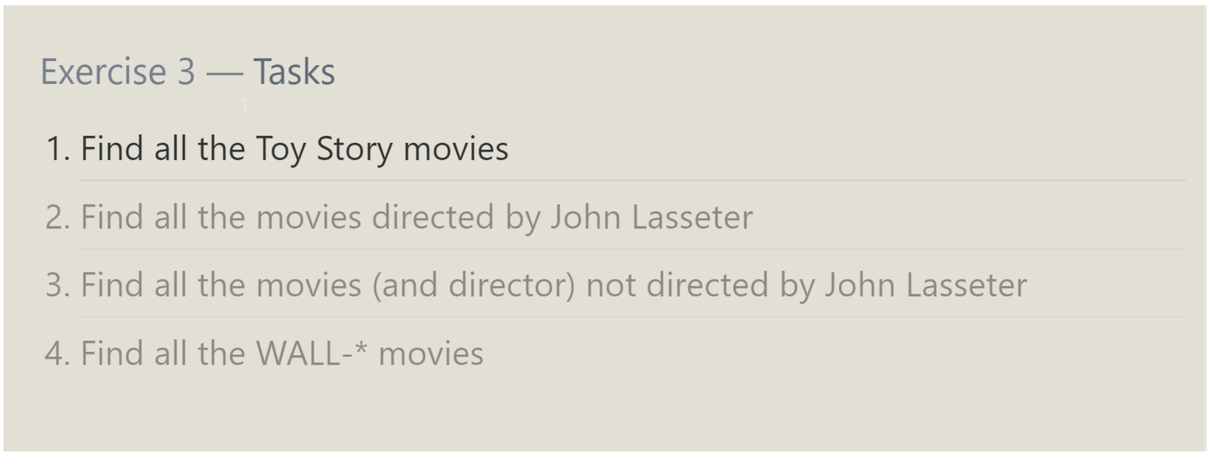
```
SELECT title,year FROM movies WHERE id<6
```

Exercise 3

SQLBolt - Learn SQL - SQL Lesson 3: Queries with constraints (Pt. 2)

When writing WHERE clauses with columns containing text data, SQL supports a number of useful operators to do things like case-insensitive string comparison and wildcard pattern matching.

https://sqlbolt.com/lesson/select_queries_with_constraints_pt_2



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

3(i). Find all the Toy Story movies

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

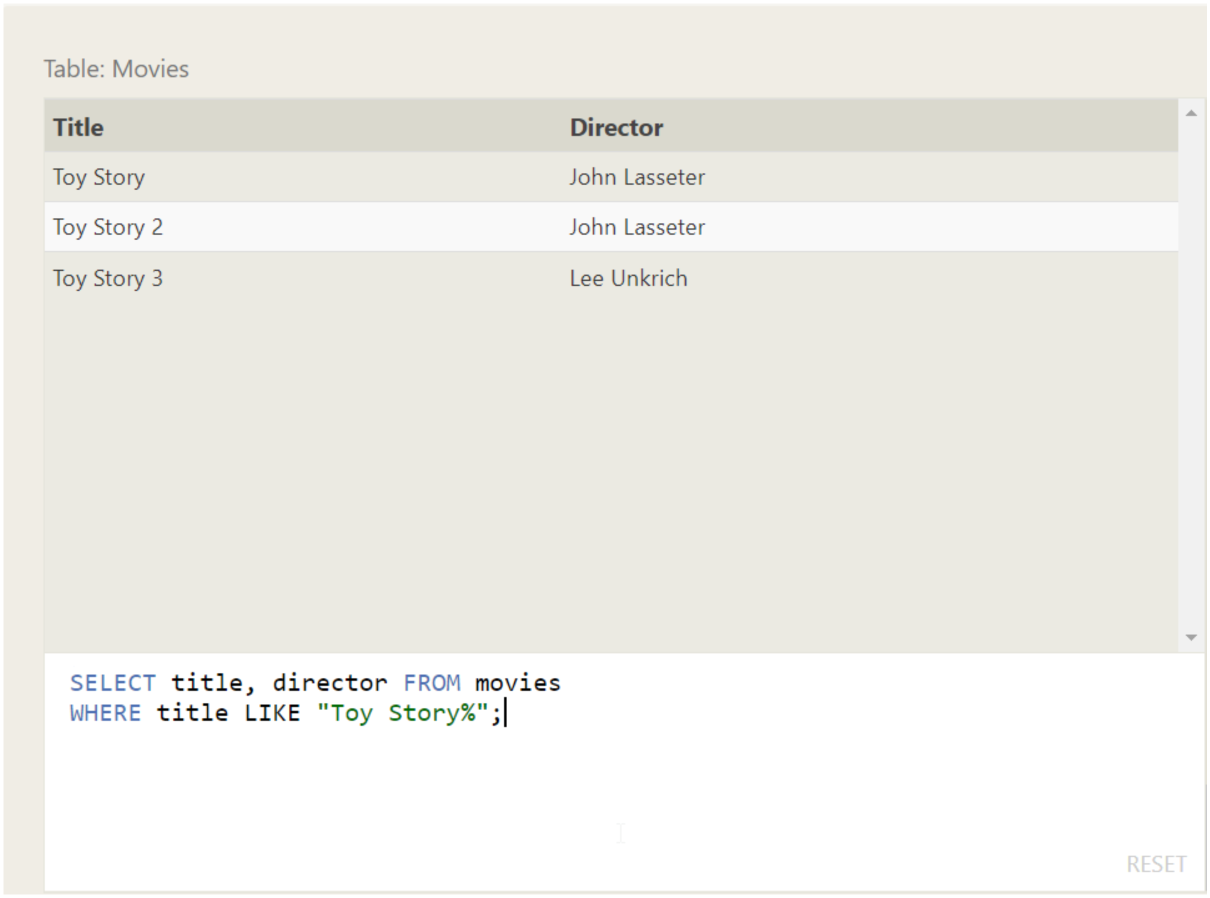


Table: Movies

Title	Director
Toy Story	John Lasseter
Toy Story 2	John Lasseter
Toy Story 3	Lee Unkrich

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

3(ii). Find all the movies directed by John Lasseter

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

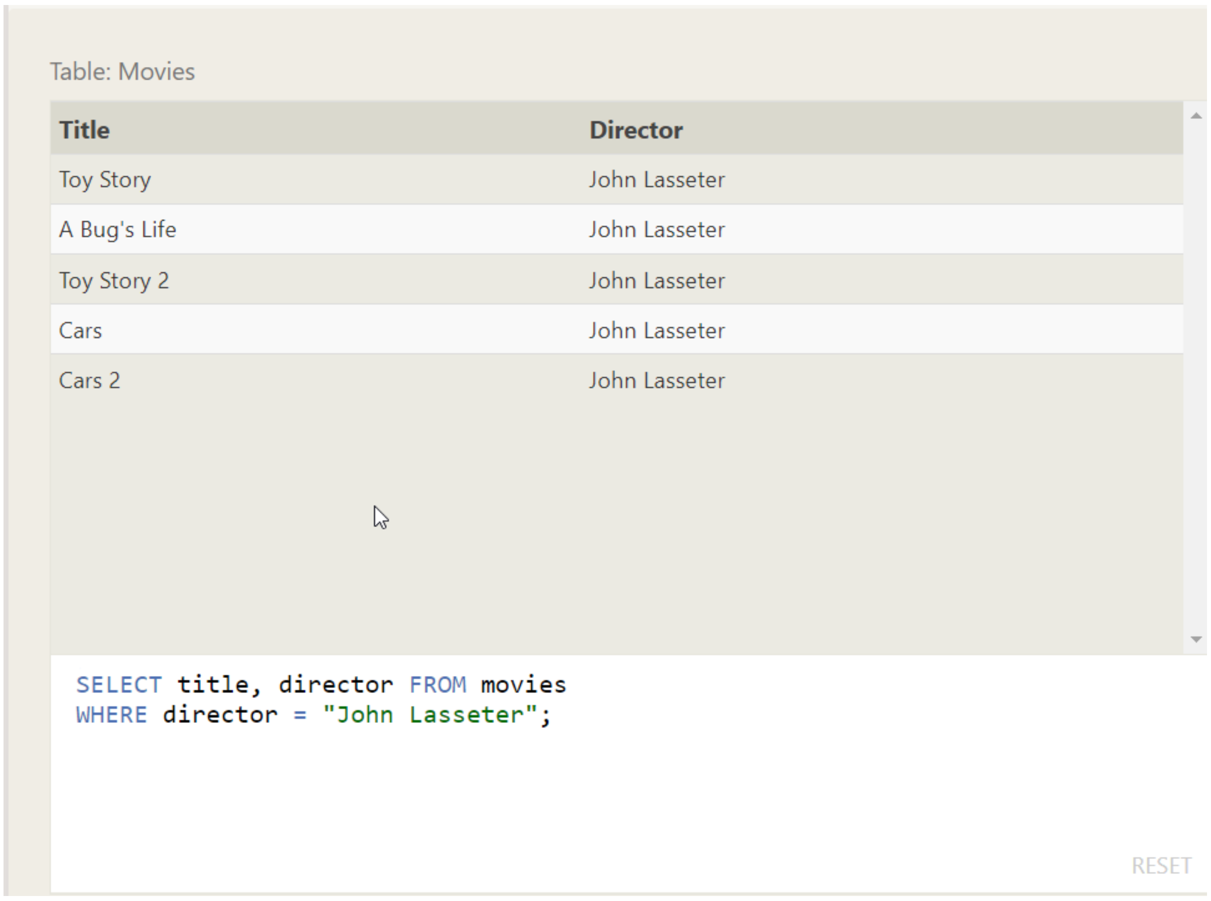


Table: Movies

Title	Director
Toy Story	John Lasseter
A Bug's Life	John Lasseter
Toy Story 2	John Lasseter
Cars	John Lasseter
Cars 2	John Lasseter

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

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

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

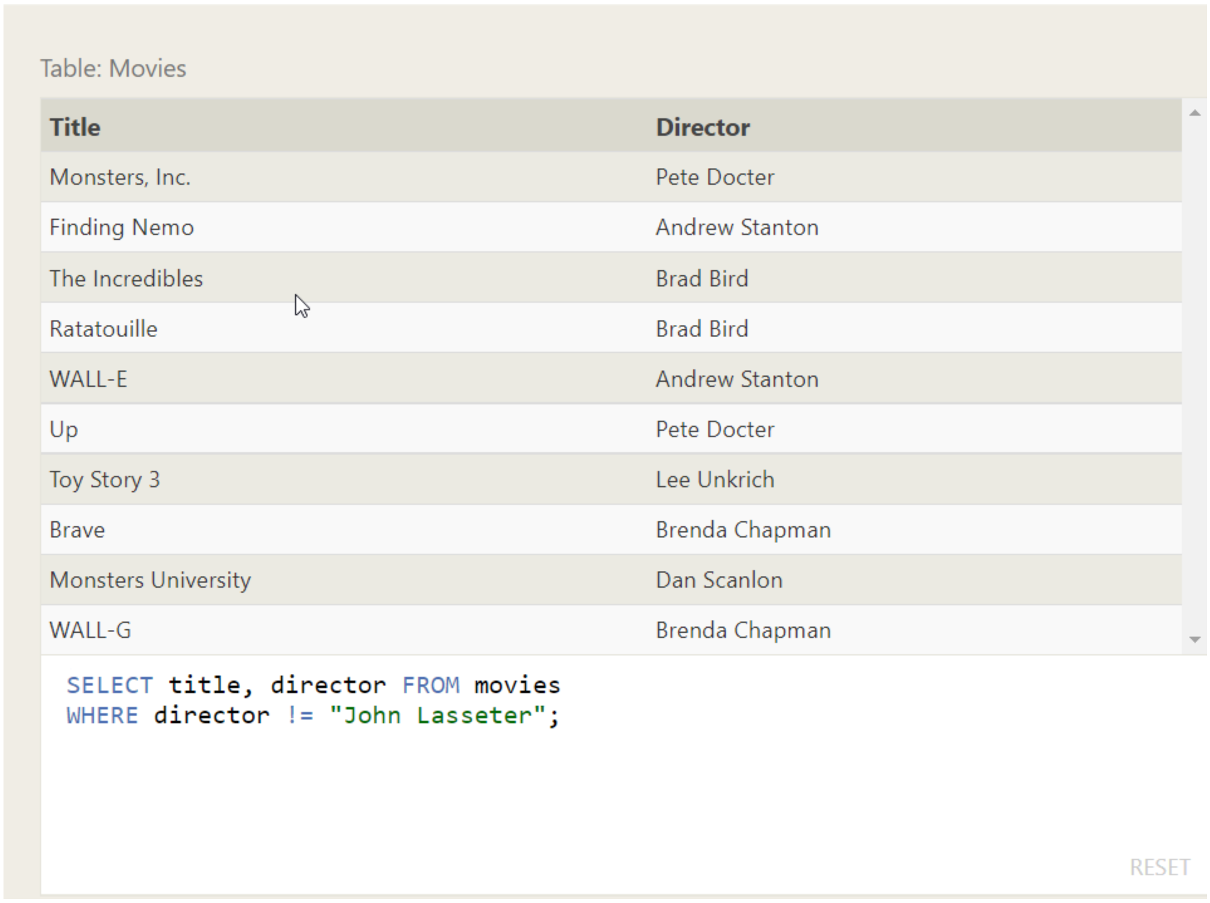


Table: Movies

Title	Director
Monsters, Inc.	Pete Docter
Finding Nemo	Andrew Stanton
The Incredibles	Brad Bird
Ratatouille	Brad Bird
WALL-E	Andrew Stanton
Up	Pete Docter
Toy Story 3	Lee Unkrich
Brave	Brenda Chapman
Monsters University	Dan Scanlon
WALL-G	Brenda Chapman

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

3(iv).Find all the WALL-* movies

```
SELECT * FROM movies
WHERE title LIKE "WALL-_";
```

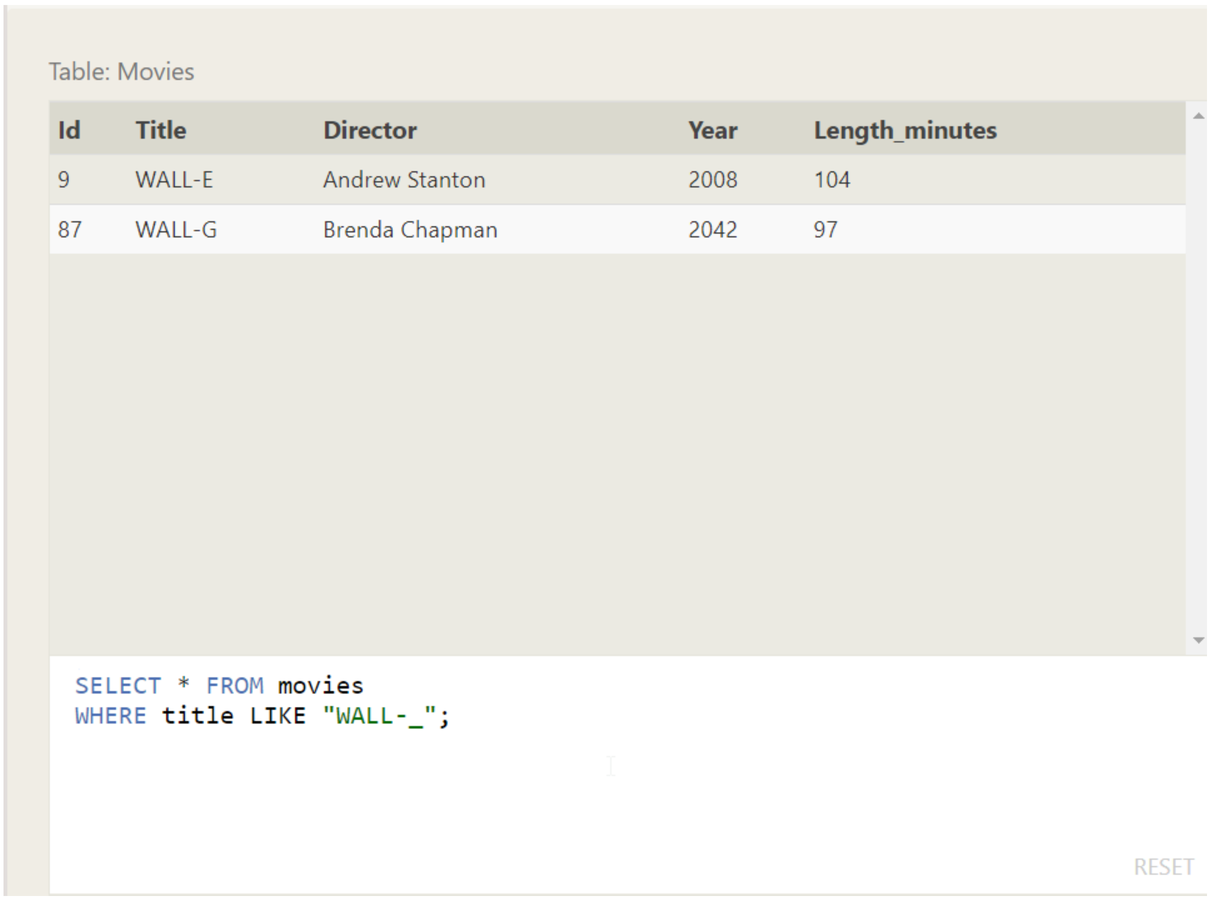


Table: Movies

Id	Title	Director	Year	Length_minutes
9	WALL-E	Andrew Stanton	2008	104
87	WALL-G	Brenda Chapman	2042	97

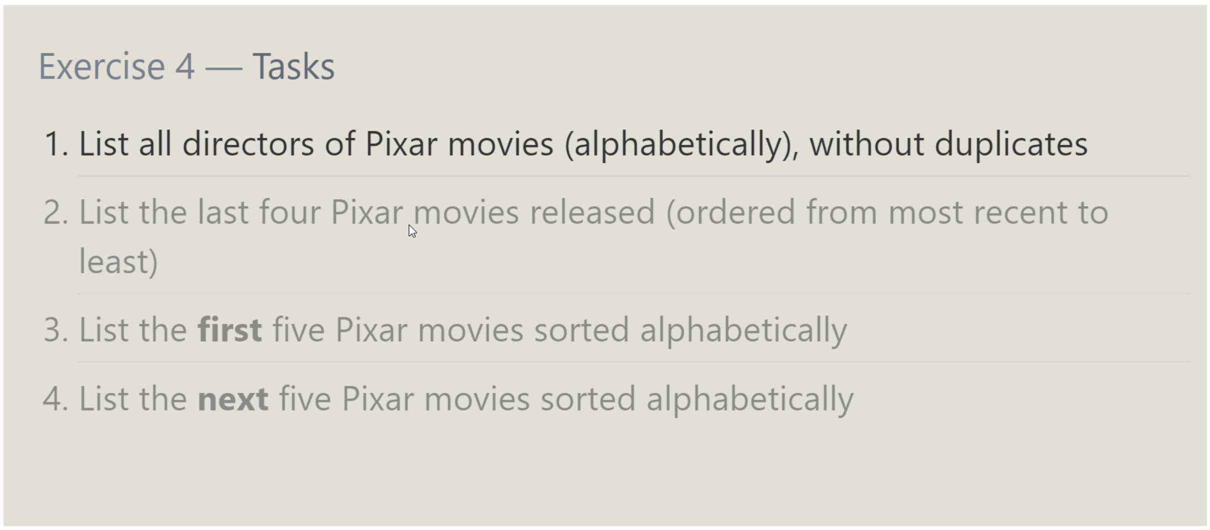
```
SELECT * FROM movies
WHERE title LIKE "WALL-_";
```

Exercise 4

SQLBolt - Learn SQL - SQL Lesson 4: Filtering and sorting Query results

Even though the data in a database may be unique, the results of any particular query may not be - take our Movies table for example, many different movies can be released the same year. In such cases, SQL provides a convenient way to discard rows that have a duplicate column value by using the DISTINCT

https://sqlbolt.com/lesson/filtering_sorting_query_results


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

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

```
SELECT DISTINCT director FROM movies
ORDER BY director ASC;
```

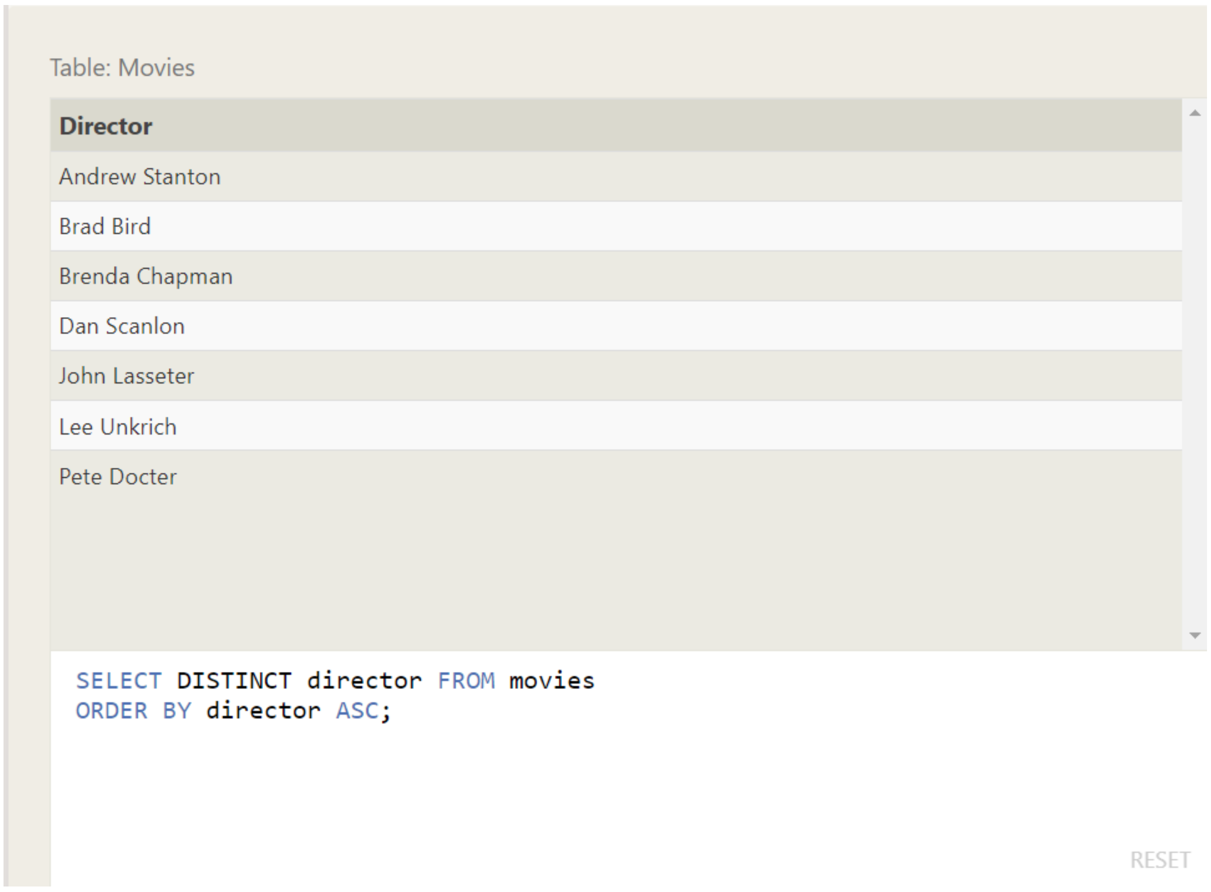


Table: Movies

Director
Andrew Stanton
Brad Bird
Brenda Chapman
Dan Scanlon
John Lasseter
Lee Unkrich
Pete Docter

```
SELECT DISTINCT director FROM movies
ORDER BY director ASC;
```

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

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

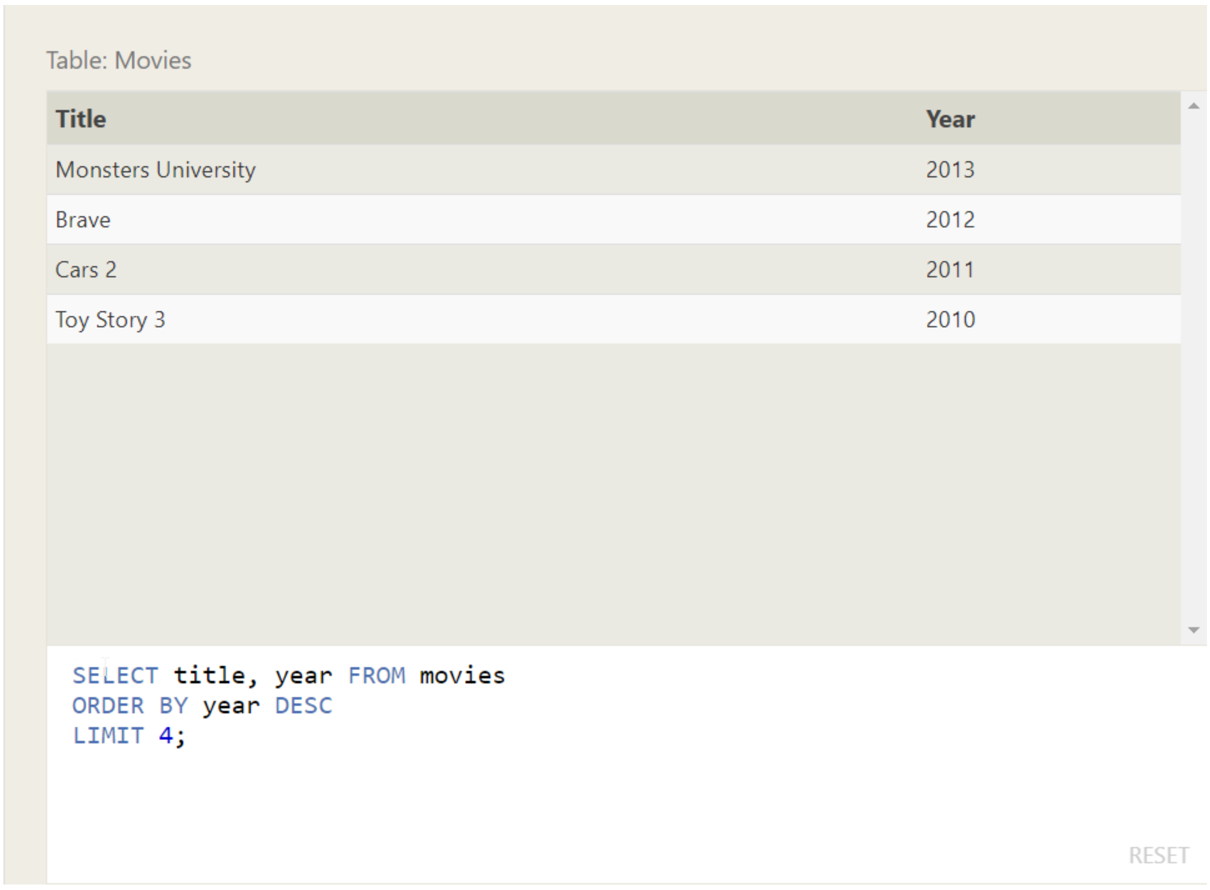


Table: Movies

Title	Year
Monsters University	2013
Brave	2012
Cars 2	2011
Toy Story 3	2010

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

4(iii).List the first five Pixar movies sorted alphabetically

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

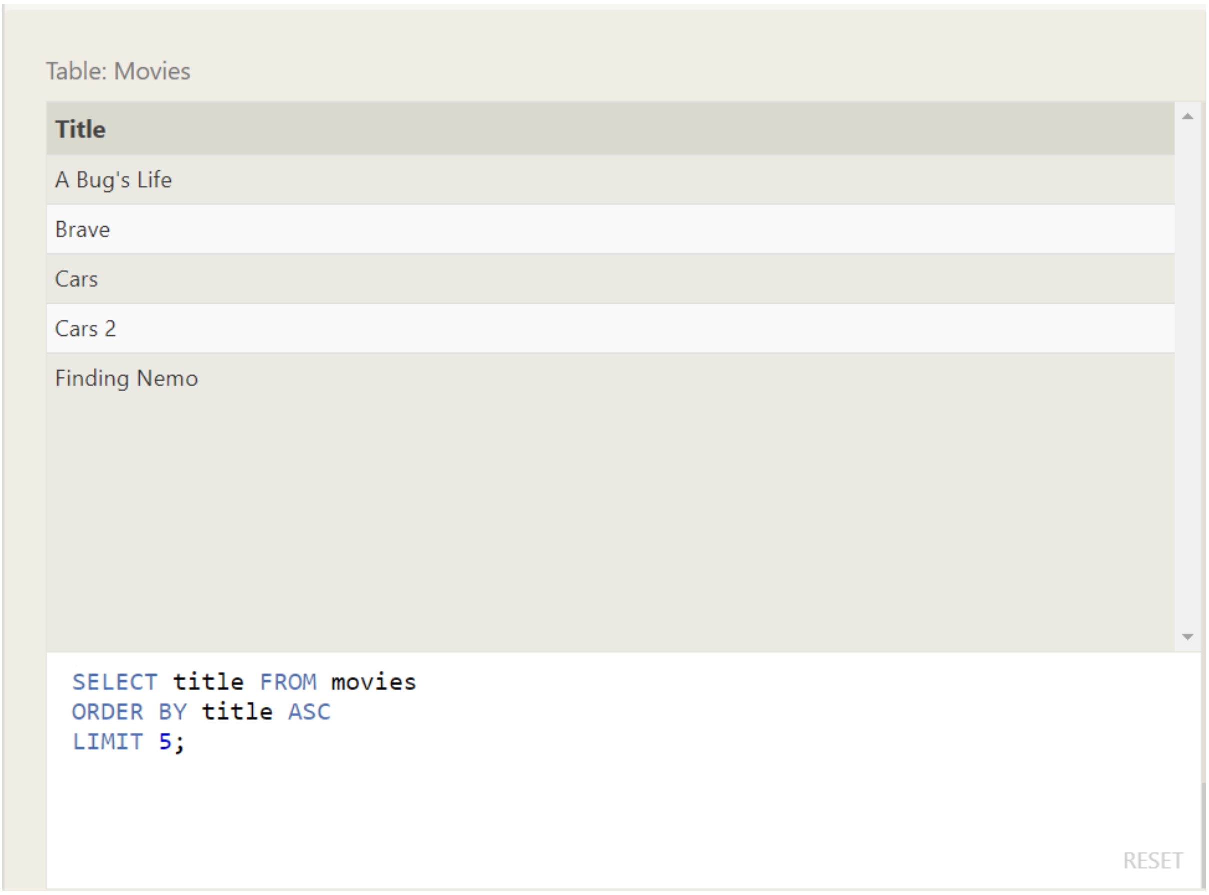


Table: Movies

Title
A Bug's Life
Brave
Cars
Cars 2
Finding Nemo

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

4(iv). List the next five Pixar movies sorted alphabetically

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

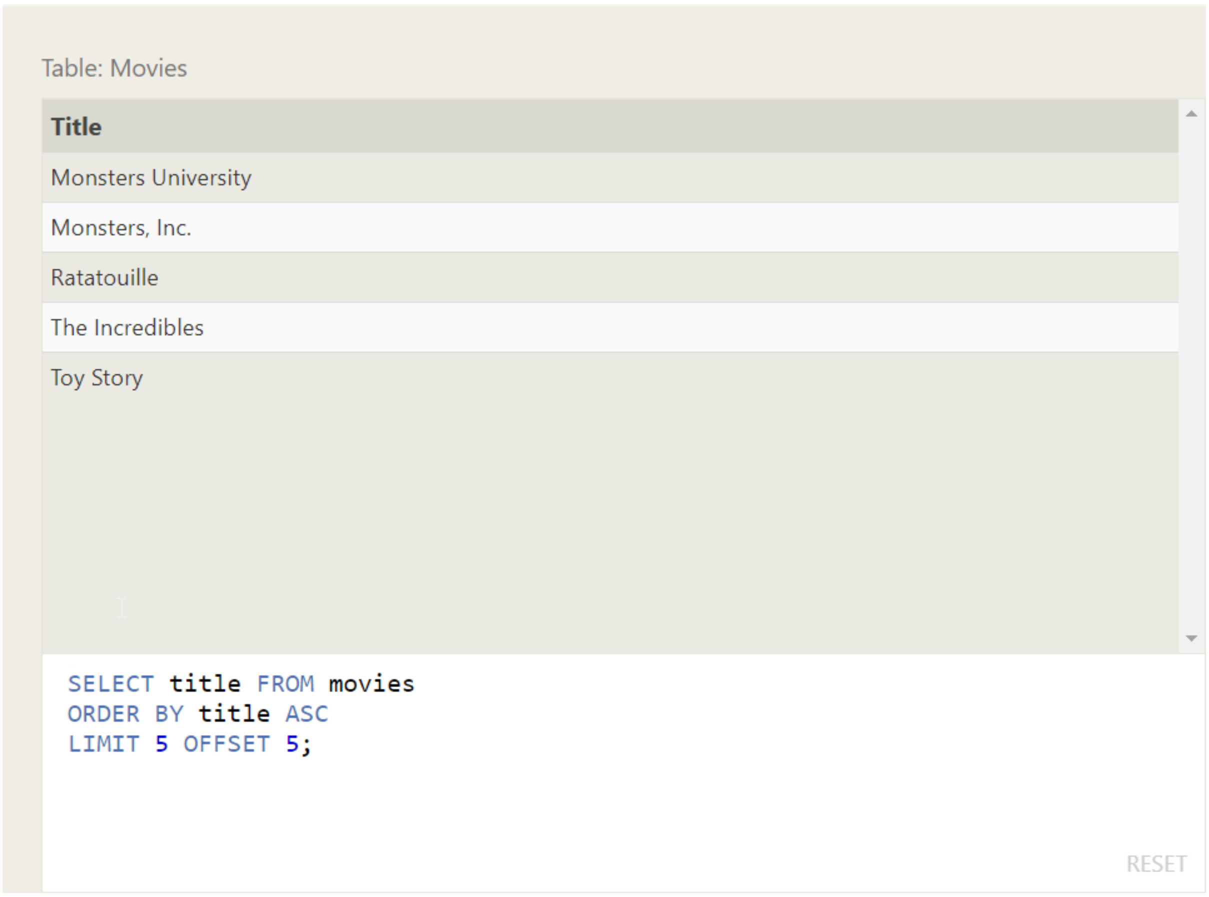


Table: Movies

Title
Monsters University
Monsters, Inc.
Ratatouille
The Incredibles
Toy Story

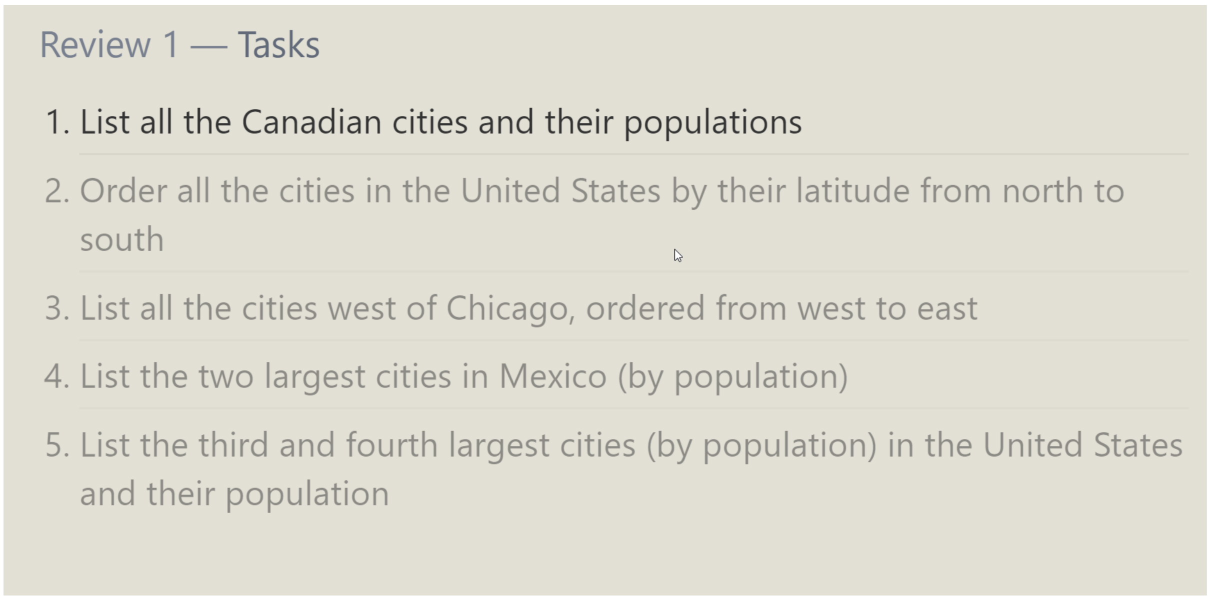
```
SELECT title FROM movies
ORDER BY title ASC
LIMIT 5 OFFSET 5;
```

SQL Review (Exercise 5)

SQLBolt - Learn SQL - SQL Review: Simple SELECT Queries

You've done a good job getting to this point! Now that you've gotten a taste of how to write a basic query, you need to practice writing queries that solve actual problems. SELECT column, another_column, ... FROM mytable WHERE condition(s) ORDER BY column ASC/DESC LIMIT num_limit OFFSET

https://sqlbolt.com/lesson/select_queries_review



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

5(i). List all the Canadian cities and their populations

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

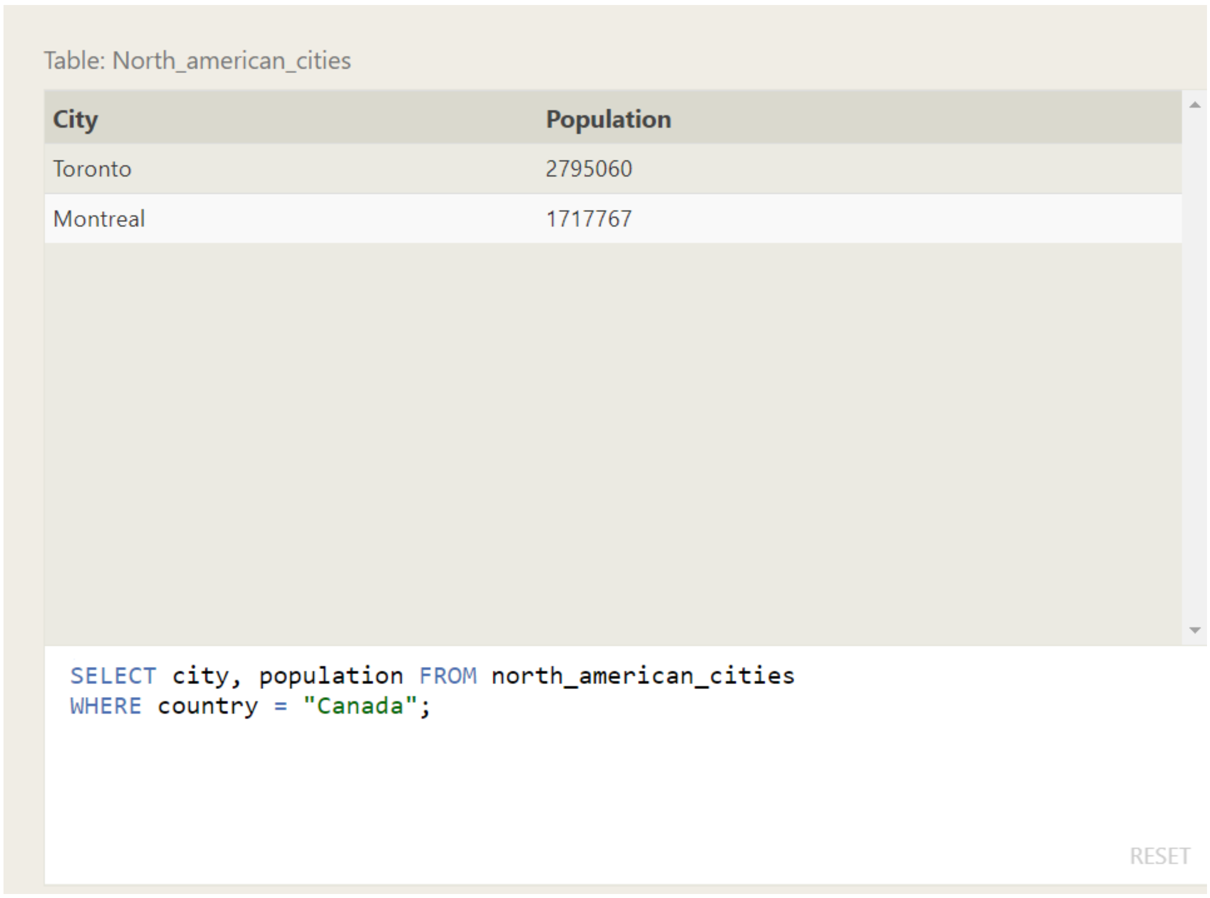


Table: North_american_cities

City	Population
Toronto	2795060
Montreal	1717767

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

5(ii).Order all the cities in the United States by their latitude from north to south

```sql
SELECT city, latitude FROM north_american_cities
WHERE country = "United States"
ORDER BY latitude DESC;
```

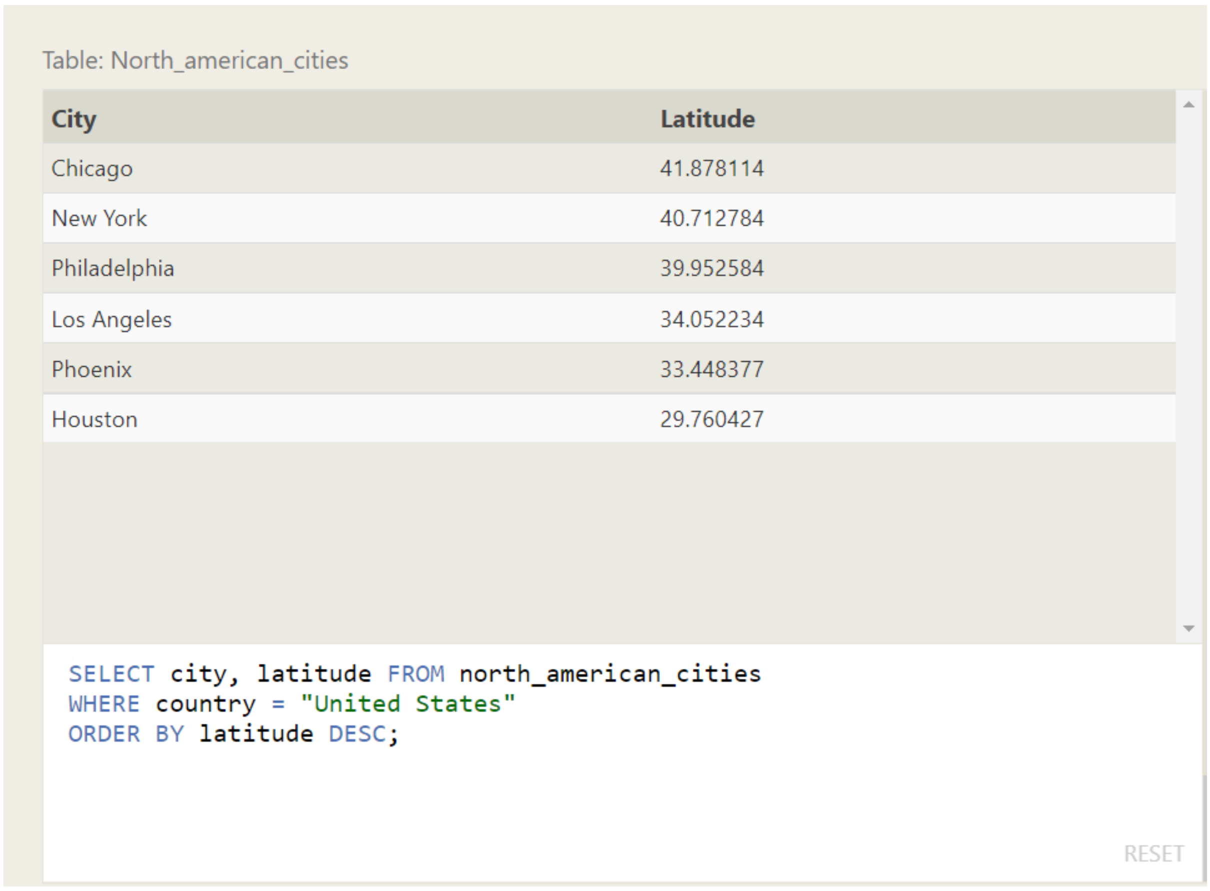


Table: North_american_cities

City	Latitude
Chicago	41.878114
New York	40.712784
Philadelphia	39.952584
Los Angeles	34.052234
Phoenix	33.448377
Houston	29.760427

```sql
SELECT city, latitude FROM north_american_cities
WHERE country = "United States"
ORDER BY latitude DESC;
```

5(iii). List all the cities west of Chicago, ordered from west to east

```sql
SELECT city, longitude FROM north_american_cities
WHERE longitude < -87.629798
ORDER BY longitude ASC;
```

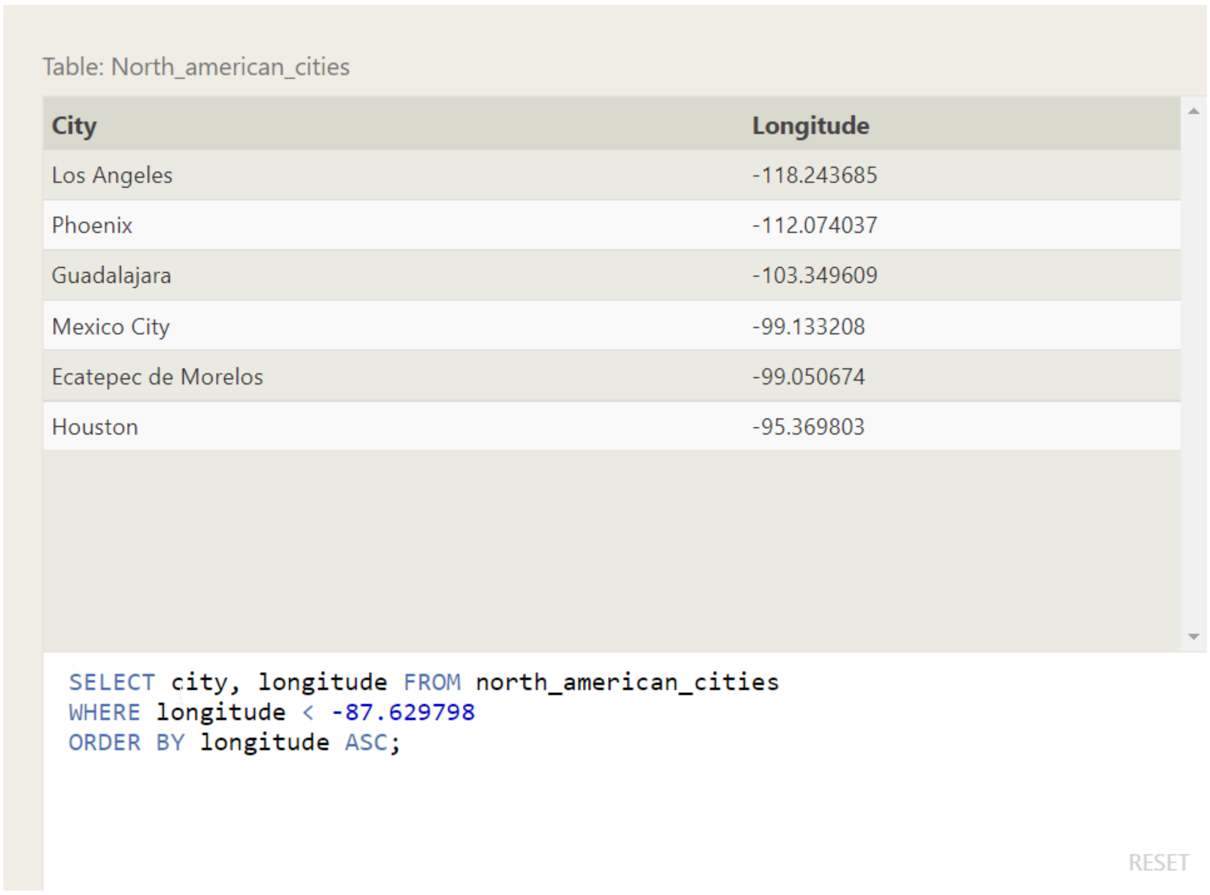


Table: North_american_cities

City	Longitude
Los Angeles	-118.243685
Phoenix	-112.074037
Guadalajara	-103.349609
Mexico City	-99.133208
Ecatepec de Morelos	-99.050674
Houston	-95.369803

```
SELECT city, longitude FROM north_american_cities
WHERE longitude < -87.629798
ORDER BY longitude ASC;
```

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

```
SELECT city, population FROM north_american_cities
WHERE country LIKE "Mexico"
ORDER BY population DESC
LIMIT 2;
```

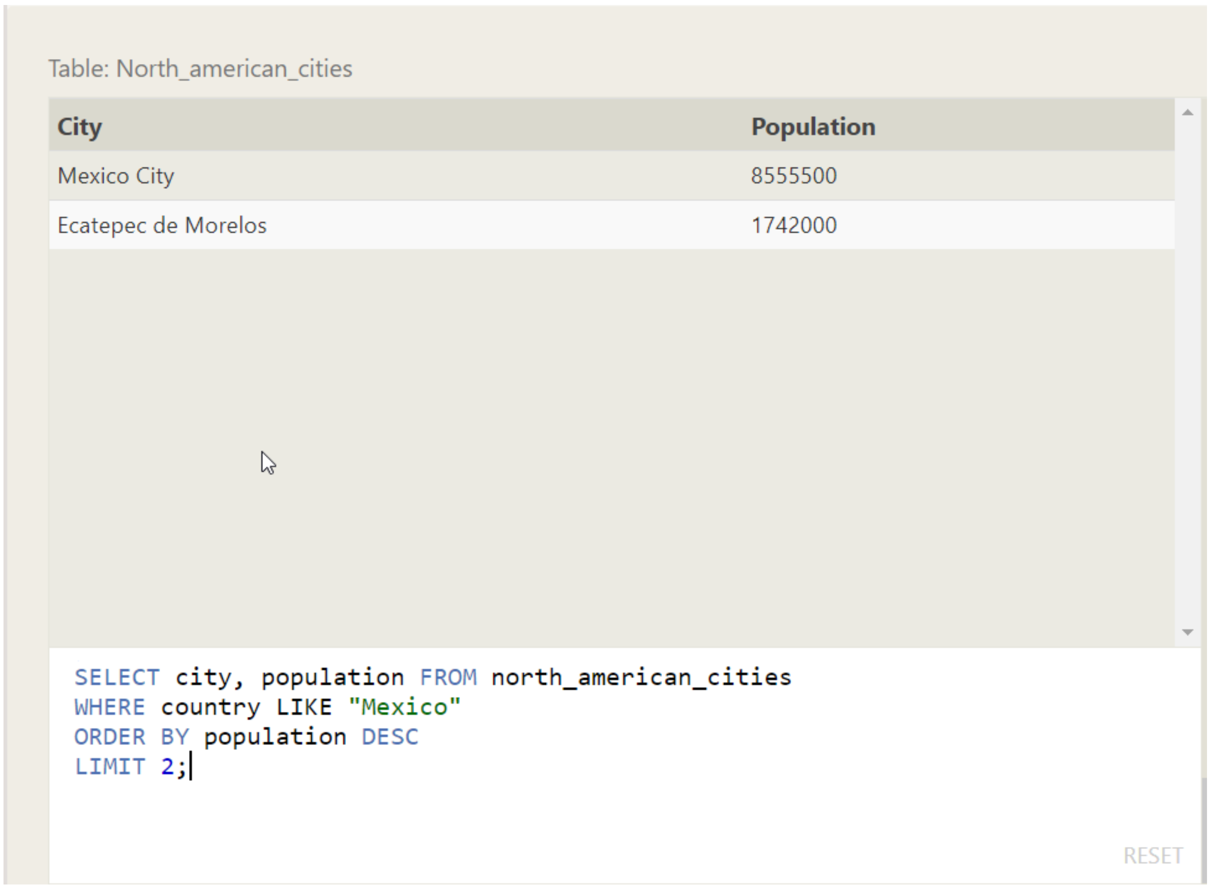


Table: North_american_cities

City	Population
Mexico City	8555500
Ecatepec de Morelos	1742000

```
SELECT city, population FROM north_american_cities
WHERE country LIKE "Mexico"
ORDER BY population DESC
LIMIT 2;
```

5(v).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 LIKE "United States"
ORDER BY population DESC
LIMIT 2 OFFSET 2;
```

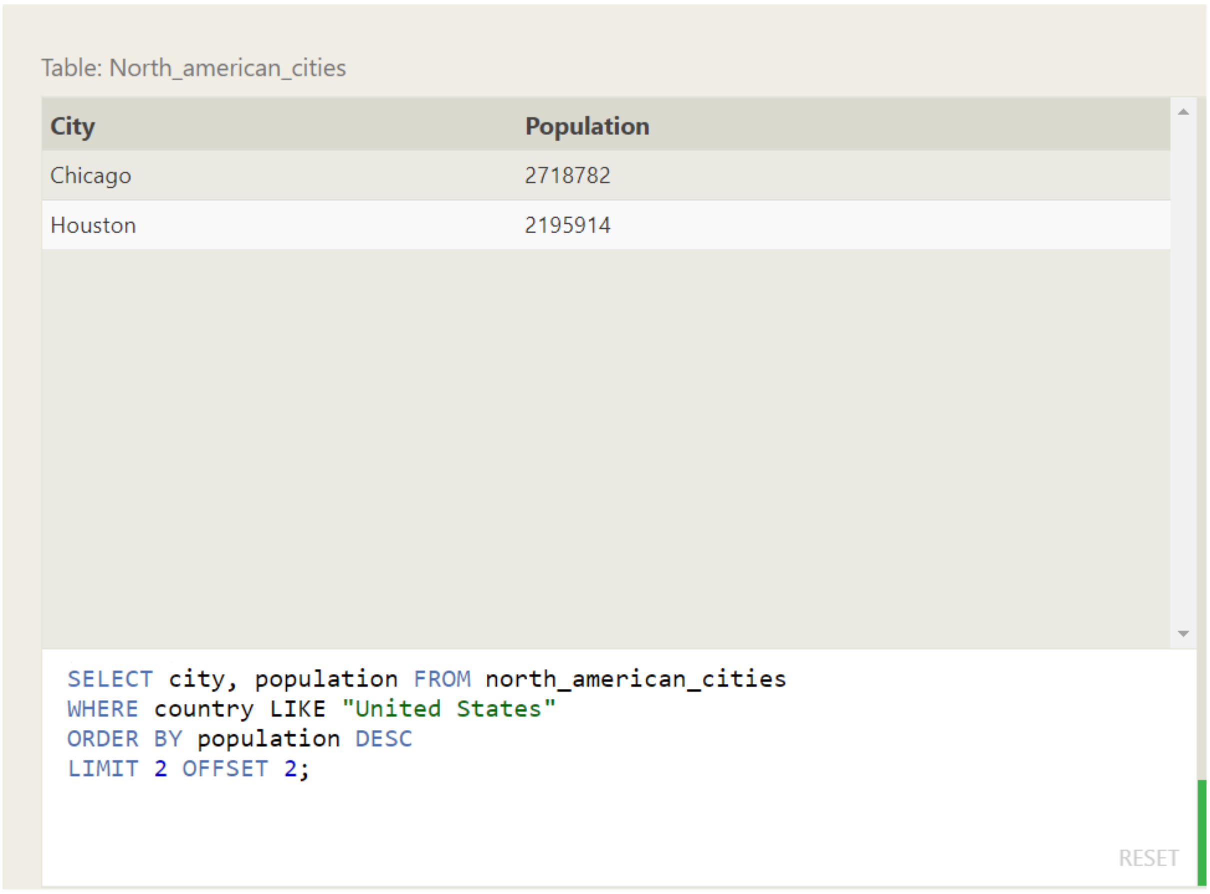


Table: North_american_cities

City	Population
Chicago	2718782
Houston	2195914

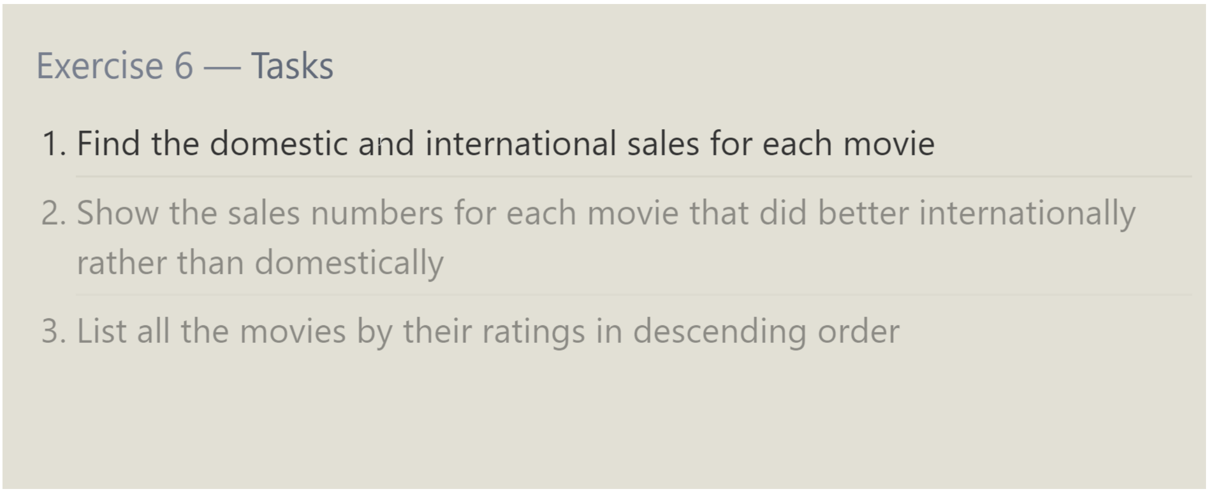
```
SELECT city, population FROM north_american_cities
WHERE country LIKE "United States"
ORDER BY population DESC
LIMIT 2 OFFSET 2;
```

Exercise 6

Multi-table queries with JOINs

Up to now, we've been working with a single table, but entity data in the real world is often broken down into pieces and stored across multiple orthogonal tables using a process known as normalization. Database normalization is useful because it minimizes duplicate data in any single table, and allows for data in the

https://sqlbolt.com/lesson/select_queries_with_joins

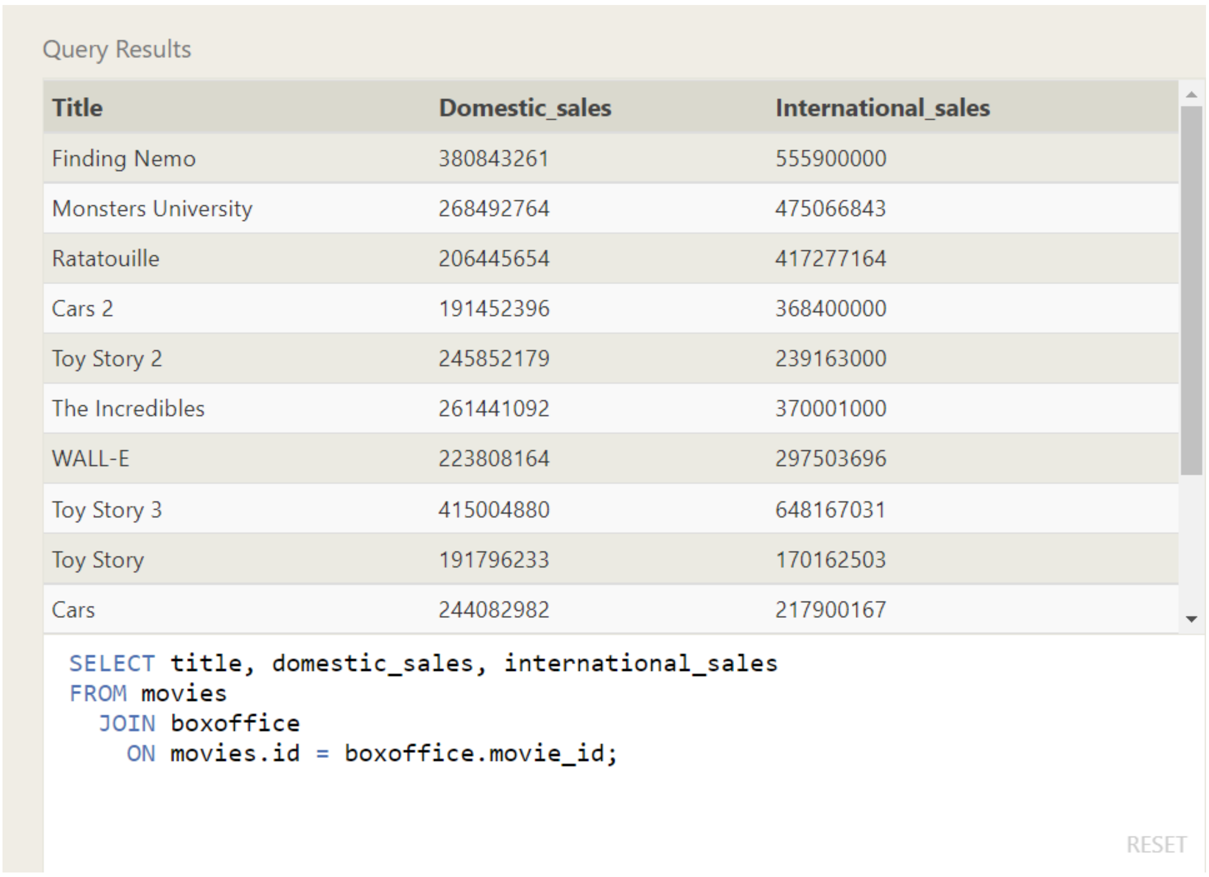


Exercise 6 — Tasks

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

6(i). Find the domestic and international sales for each movie

```
SELECT title, domestic_sales, international_sales
FROM movies
  JOIN boxoffice
    ON movies.id = boxoffice.movie_id;
```



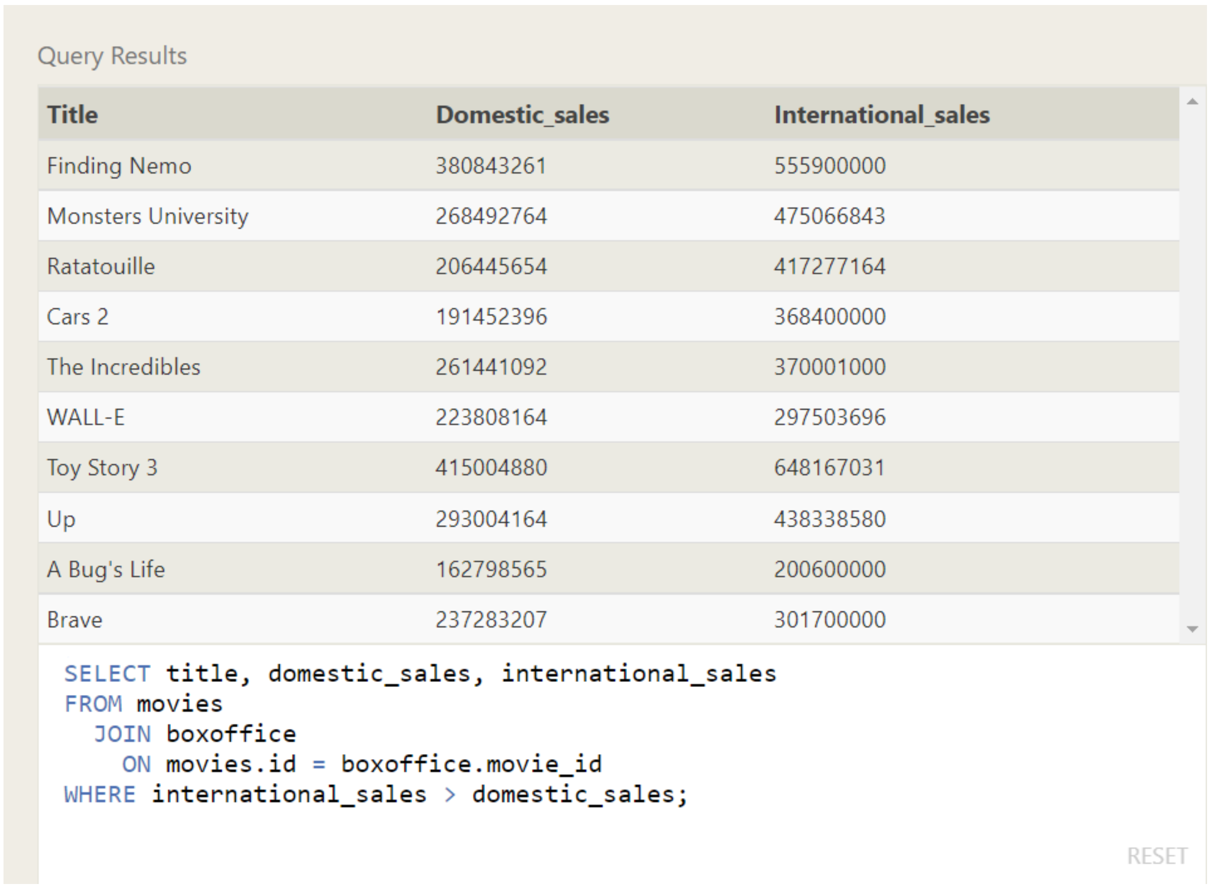
Query Results

Title	Domestic_sales	International_sales
Finding Nemo	380843261	555900000
Monsters University	268492764	475066843
Ratatouille	206445654	417277164
Cars 2	191452396	368400000
Toy Story 2	245852179	239163000
The Incredibles	261441092	370001000
WALL-E	223808164	297503696
Toy Story 3	415004880	648167031
Toy Story	191796233	170162503
Cars	244082982	217900167

```
SELECT title, domestic_sales, international_sales
FROM movies
  JOIN boxoffice
    ON movies.id = boxoffice.movie_id;
```

6(ii).Show the sales numbers for each movie that did better internationally rather than domestically

```
SELECT title, domestic_sales, international_sales
FROM movies
  JOIN boxoffice
    ON movies.id = boxoffice.movie_id
WHERE international_sales > domestic_sales;
```



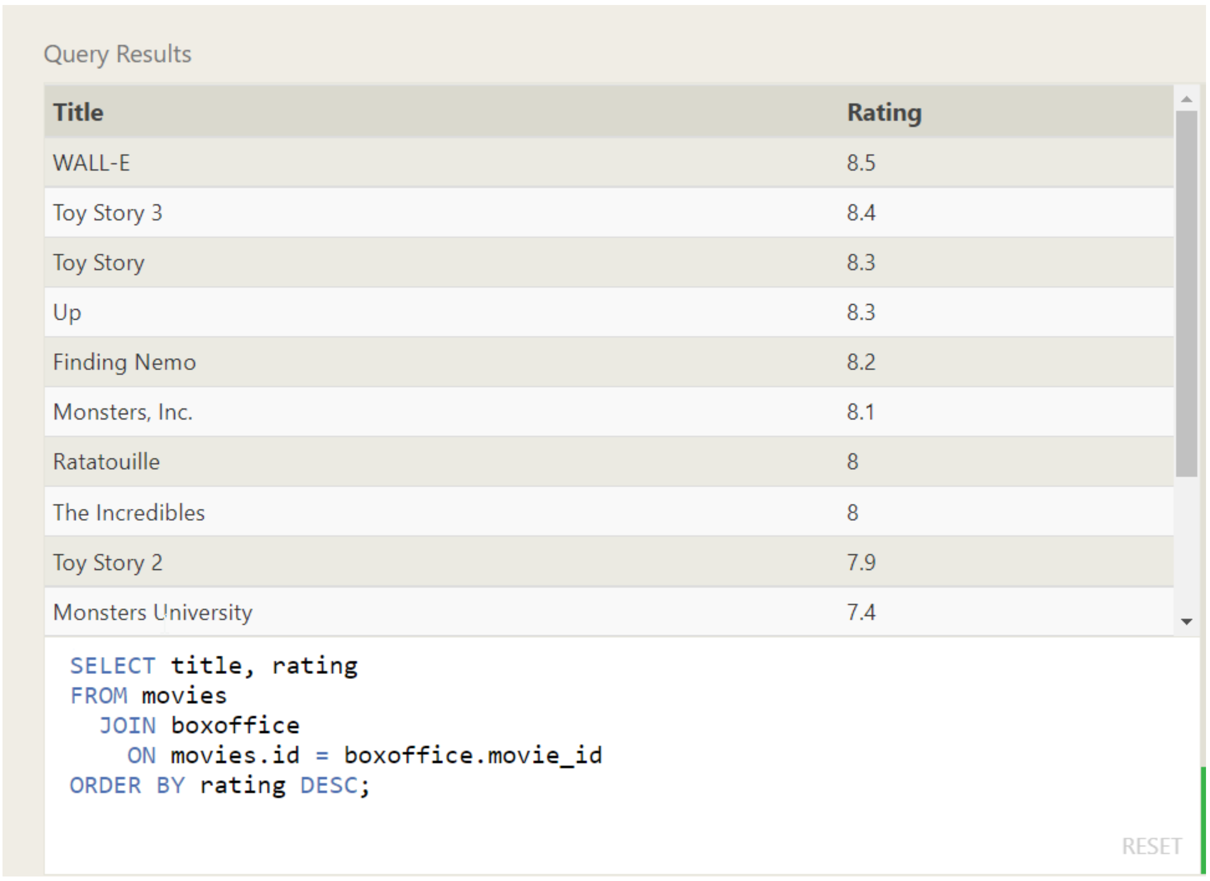
Query Results

Title	Domestic_sales	International_sales
Finding Nemo	380843261	555900000
Monsters University	268492764	475066843
Ratatouille	206445654	417277164
Cars 2	191452396	368400000
The Incredibles	261441092	370001000
WALL-E	223808164	297503696
Toy Story 3	415004880	648167031
Up	293004164	438338580
A Bug's Life	162798565	200600000
Brave	237283207	301700000

```
SELECT title, domestic_sales, international_sales
FROM movies
  JOIN boxoffice
    ON movies.id = boxoffice.movie_id
WHERE international_sales > domestic_sales;
```

6(iii).List all the movies by their ratings in descending order

```
SELECT title, rating
FROM movies
  JOIN boxoffice
    ON movies.id = boxoffice.movie_id
ORDER BY rating DESC;
```


Query Results

Title	Rating
WALL-E	8.5
Toy Story 3	8.4
Toy Story	8.3
Up	8.3
Finding Nemo	8.2
Monsters, Inc.	8.1
Ratatouille	8
The Incredibles	8
Toy Story 2	7.9
Monsters University	7.4

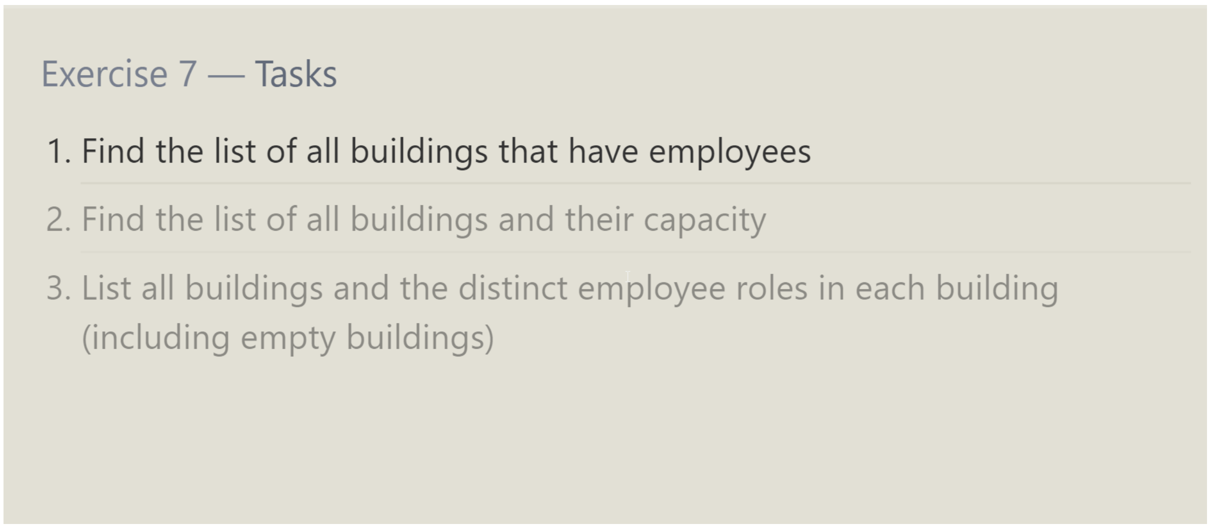
```
SELECT title, rating
FROM movies
  JOIN boxoffice
    ON movies.id = boxoffice.movie_id
ORDER BY rating DESC;
```

Exercise 7

SQLBolt - Learn SQL - SQL Lesson 7: OUTER JOINs

Depending on how you want to analyze the data, the INNER JOIN we used last lesson might not be sufficient because the resulting table only contains data that belongs in both of the tables.

https://sqlbolt.com/lesson/select_queries_with_outer_joins

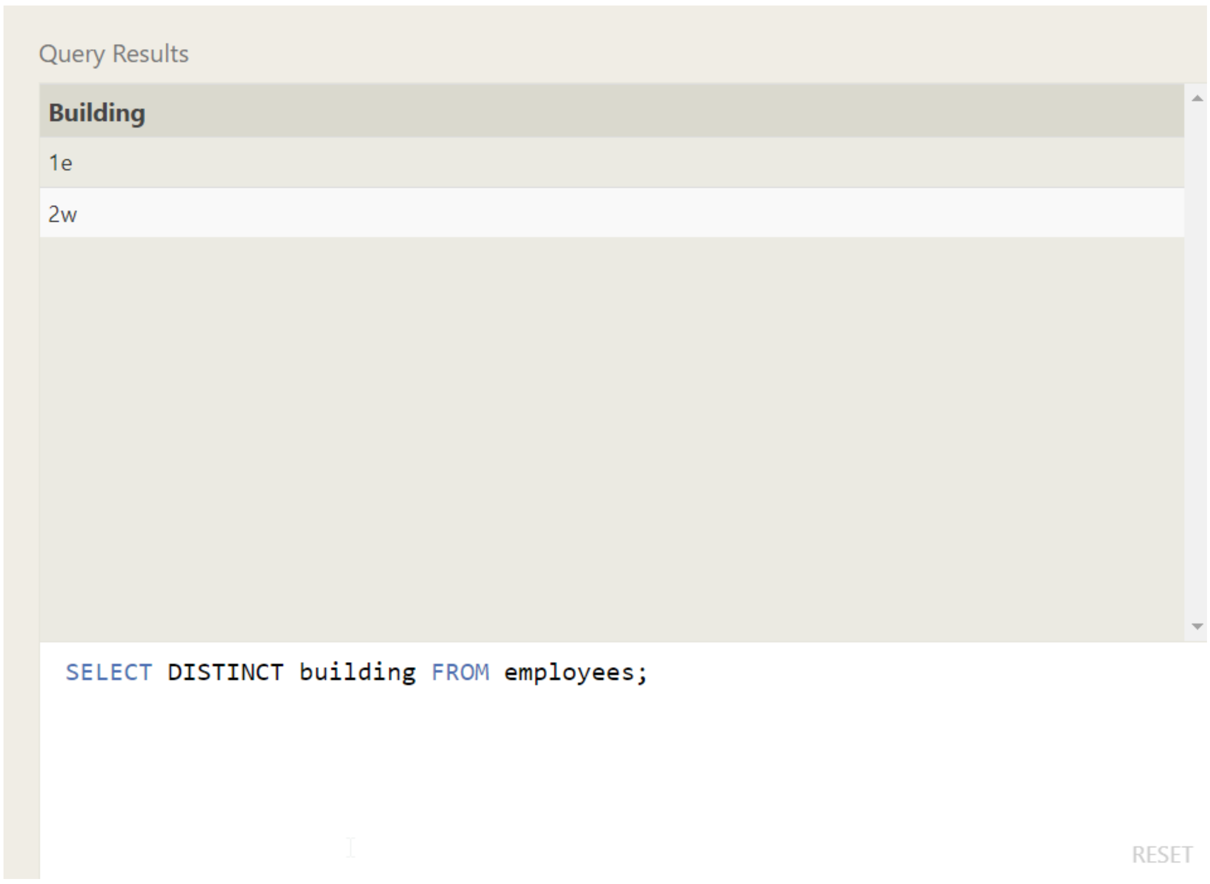


Exercise 7 — Tasks

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)

7(i).Find the list of all buildings that have employees

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



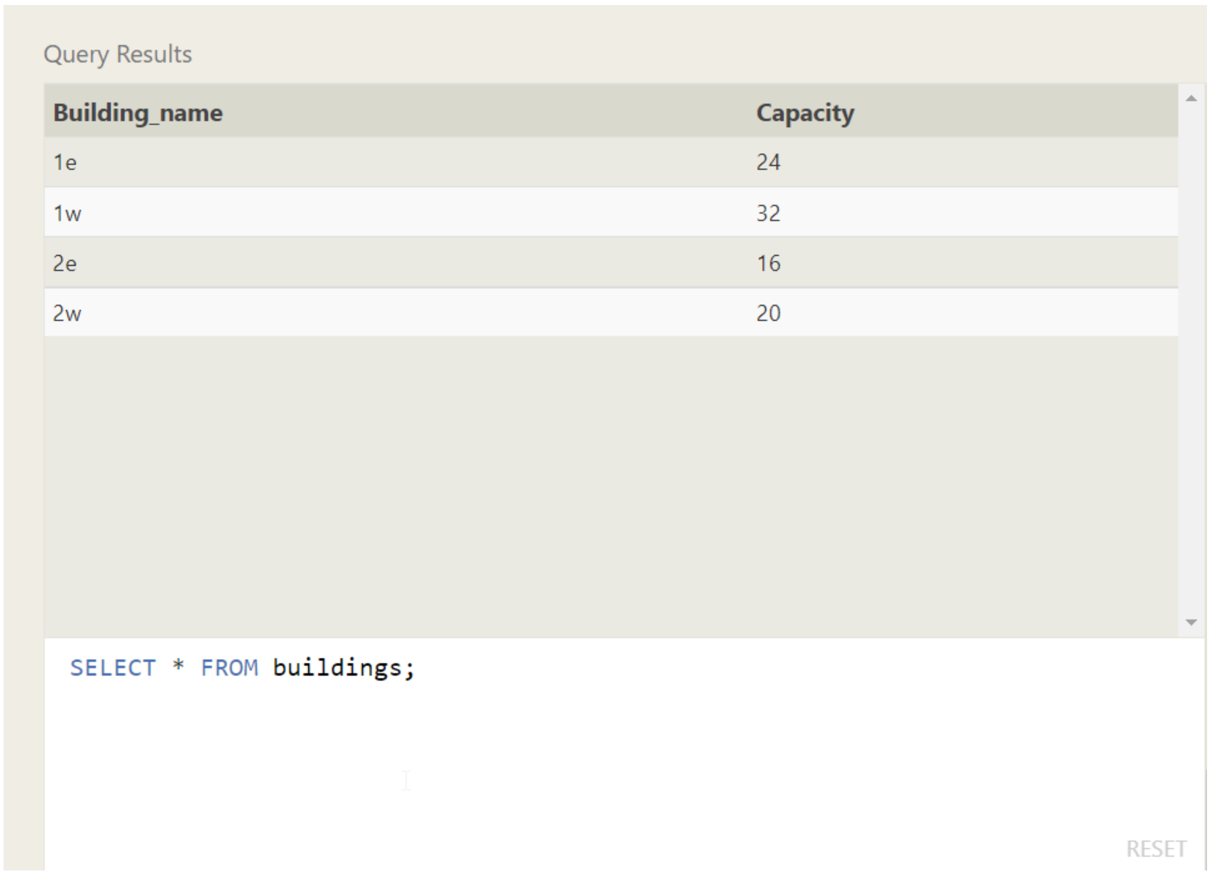
Query Results

Building
1e
2w

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

7(ii). Find the list of all buildings and their capacity

```
SELECT * FROM buildings;
```



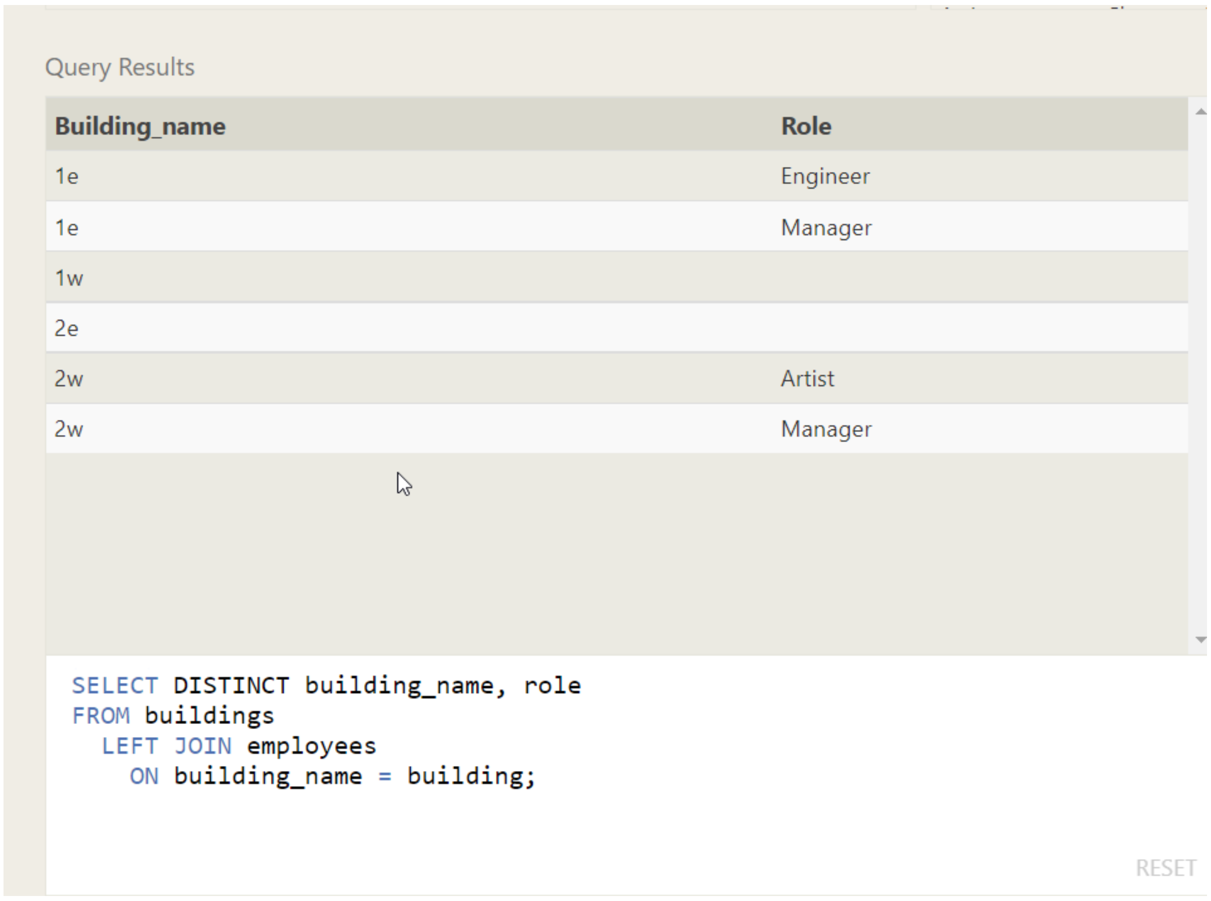
Query Results

Building_name	Capacity
1e	24
1w	32
2e	16
2w	20

```
SELECT * FROM buildings;
```

7(iii). 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;
```



Query Results

Building_name	Role
1e	Engineer
1e	Manager
1w	
2e	
2w	Artist
2w	Manager

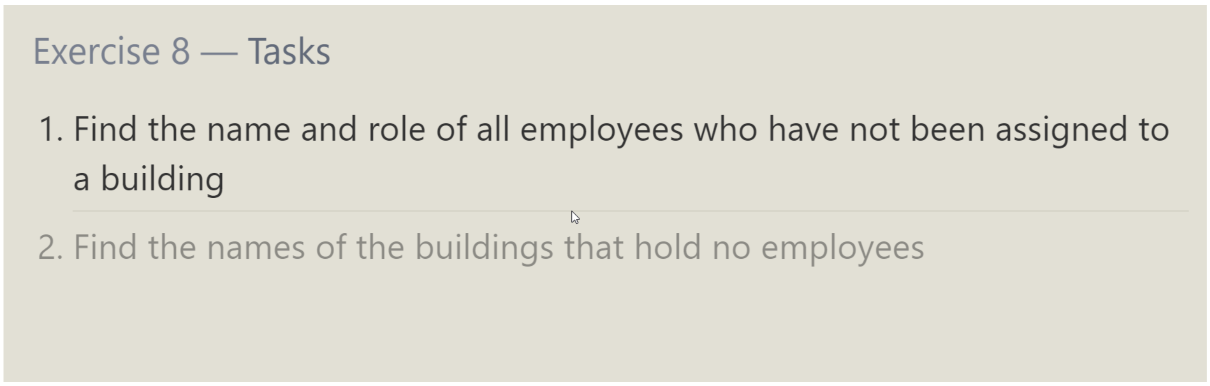
```
SELECT DISTINCT building_name, role
FROM buildings
  LEFT JOIN employees
    ON building_name = building;
```

Exercise 8

SQLBolt - Learn SQL - SQL Lesson 8: A short note on NULLs

As promised in the last lesson, we are going to quickly talk about NULL values in an SQL database. It's always good to reduce the possibility of NULL values in databases because they require special attention when constructing queries, constraints (certain functions behave differently with null values) and when

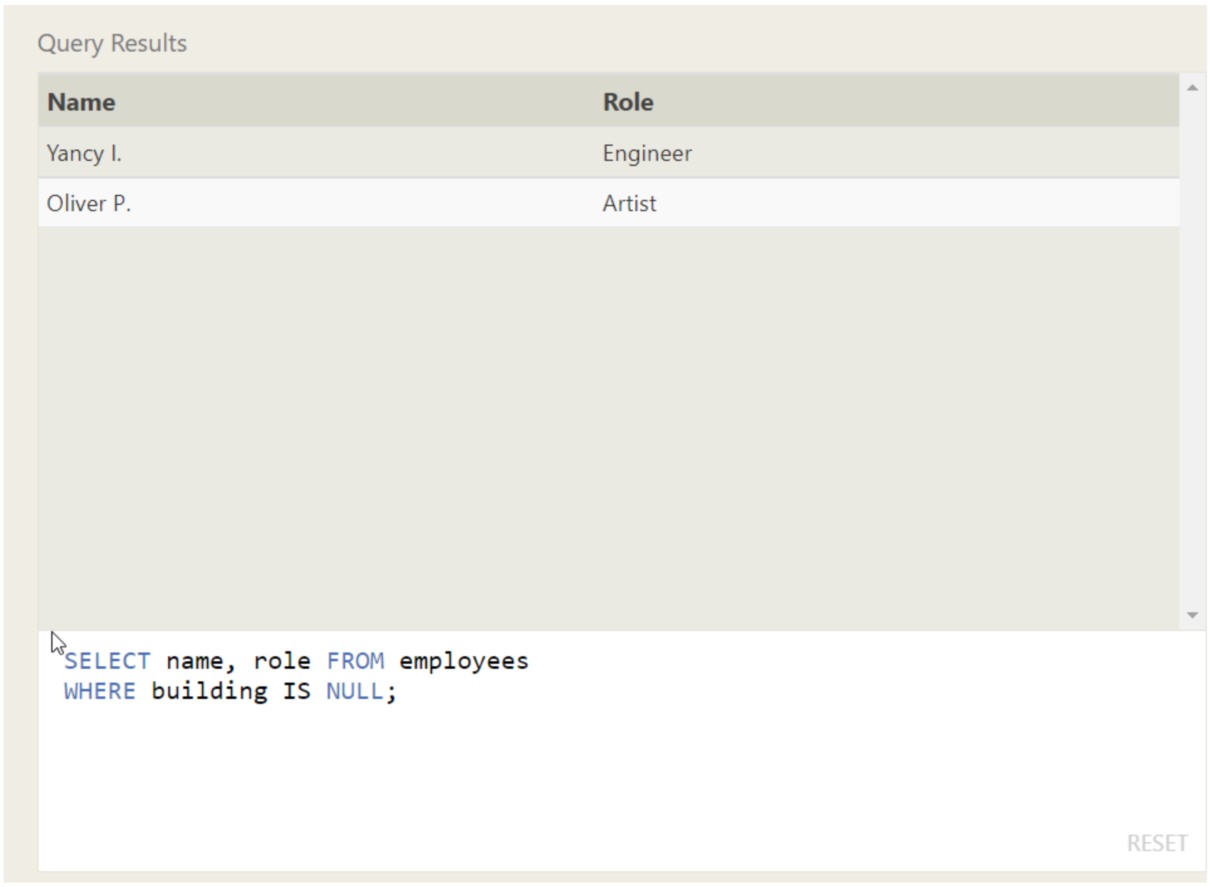
https://sqlbolt.com/lesson/select_queries_with_nulls


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

8(i).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;
```

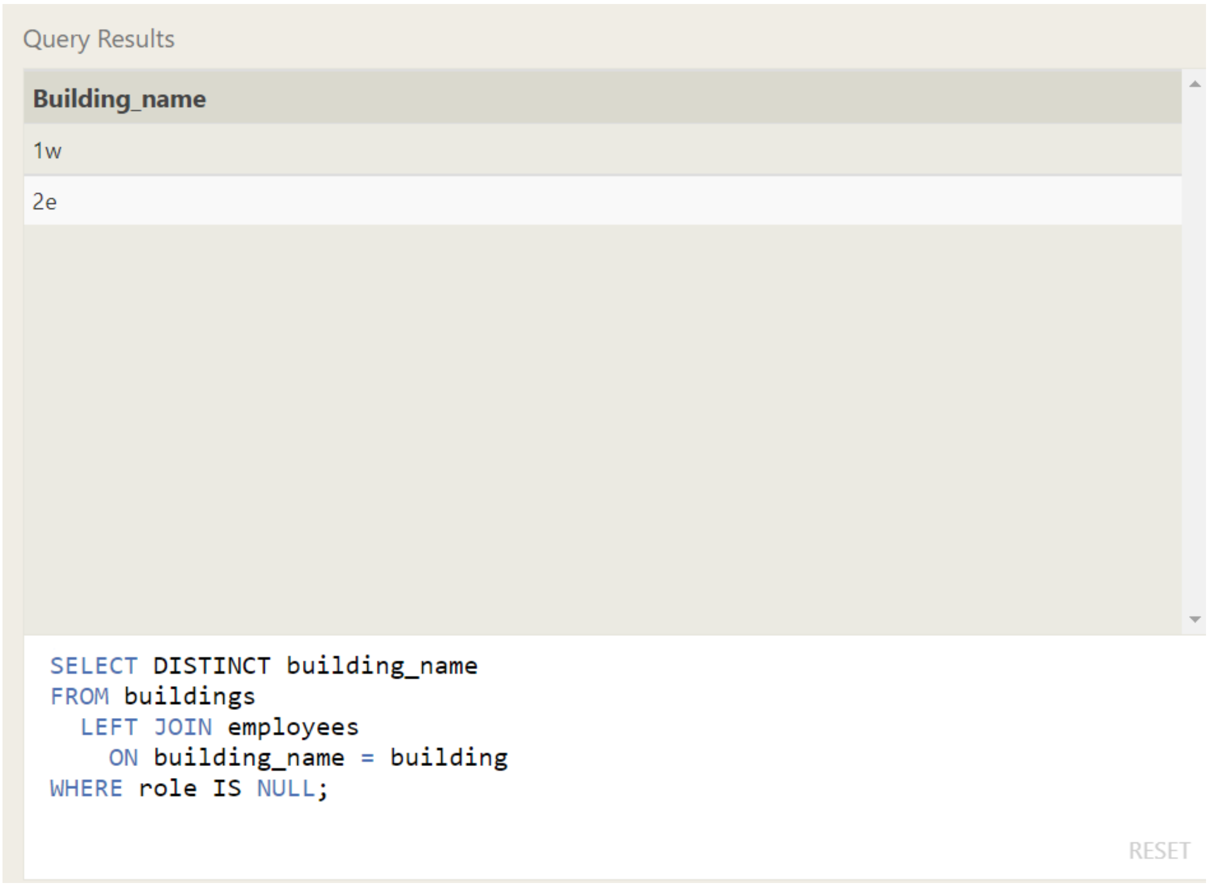

Query Results

Name	Role
Yancy I.	Engineer
Oliver P.	Artist

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

8(ii). 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;
```



Query Results

Building_name
1w
2e

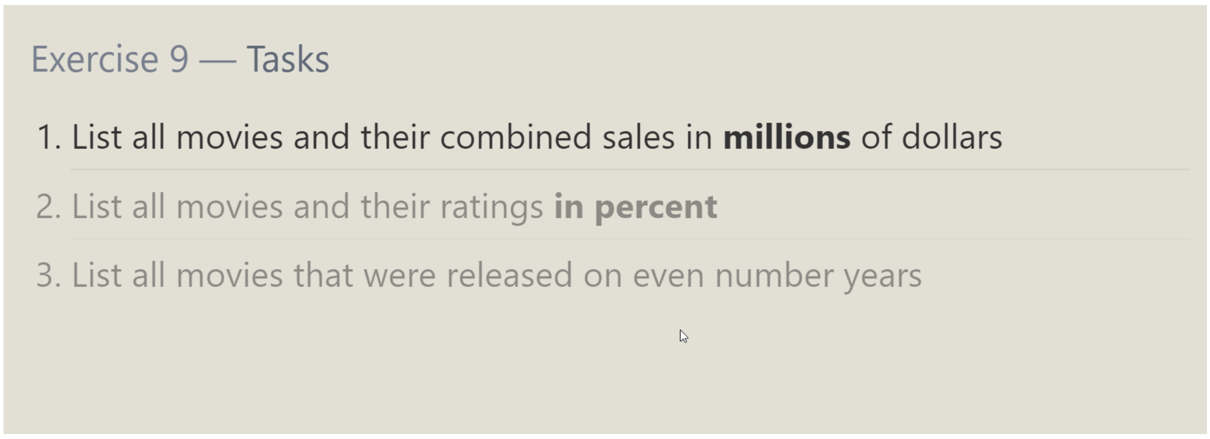
```
SELECT DISTINCT building_name
FROM buildings
  LEFT JOIN employees
    ON building_name = building
WHERE role IS NULL;
```

Exercise 9

SQLBolt - Learn SQL - SQL Lesson 9: Queries with expressions

In addition to querying and referencing raw column data with SQL, you can also use expressions to write more complex logic on column values in a query. These expressions can use mathematical and string functions along with basic arithmetic to transform values when the query is executed, as shown in this

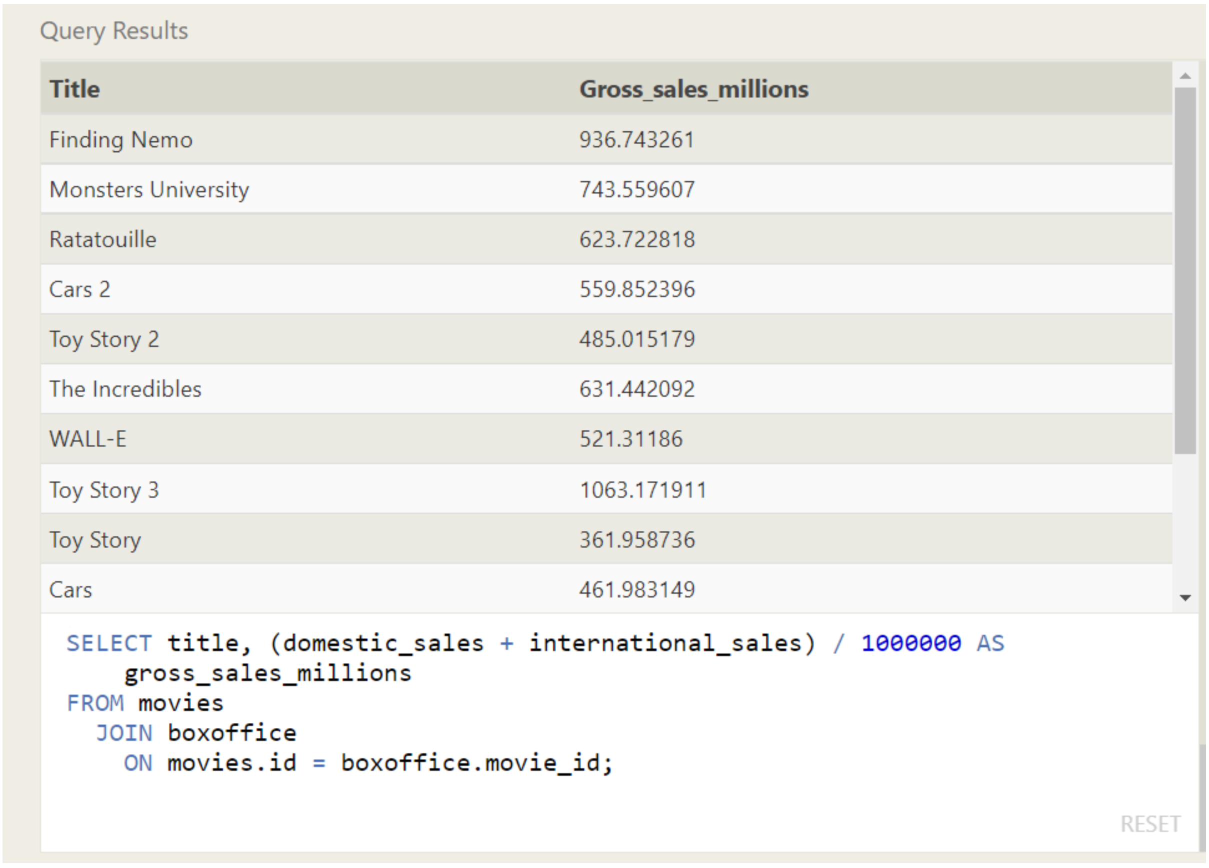
https://sqlbolt.com/lesson/select_queries_with_expressions


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

9(i).List all movies and their combined sales in millions of dollars

```
SELECT title, (domestic_sales + international_sales) / 1000000 AS gross_sales_millions
FROM movies
  JOIN boxoffice
    ON movies.id = boxoffice.movie_id;
```

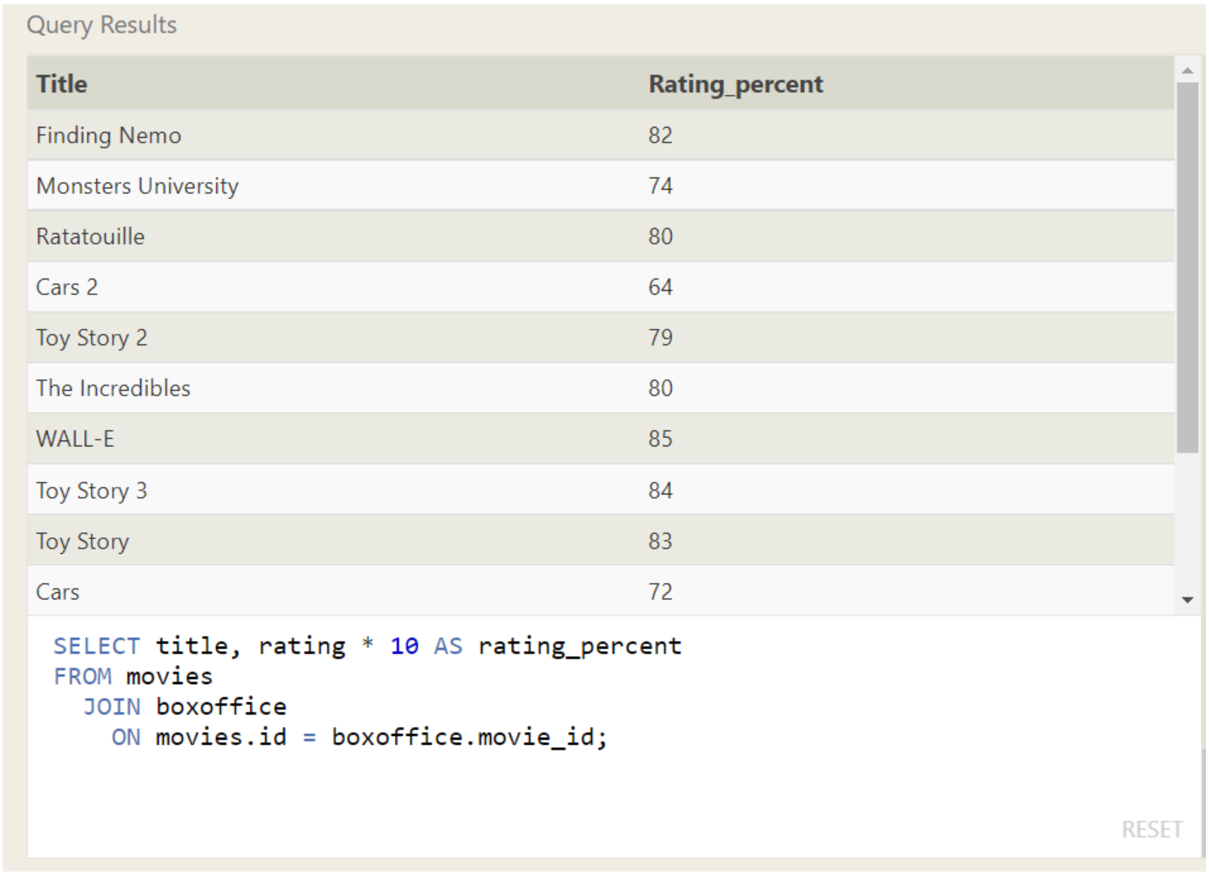


Title	Gross_sales_millions
Finding Nemo	936.743261
Monsters University	743.559607
Ratatouille	623.722818
Cars 2	559.852396
Toy Story 2	485.015179
The Incredibles	631.442092
WALL-E	521.31186
Toy Story 3	1063.171911
Toy Story	361.958736
Cars	461.983149

```
SELECT title, (domestic_sales + international_sales) / 1000000 AS
    gross_sales_millions
FROM movies
  JOIN boxoffice
    ON movies.id = boxoffice.movie_id;
```

9(ii). List all movies and their ratings in percent

```
SELECT title, rating * 10 AS rating_percent
FROM movies
  JOIN boxoffice
    ON movies.id = boxoffice.movie_id;
```

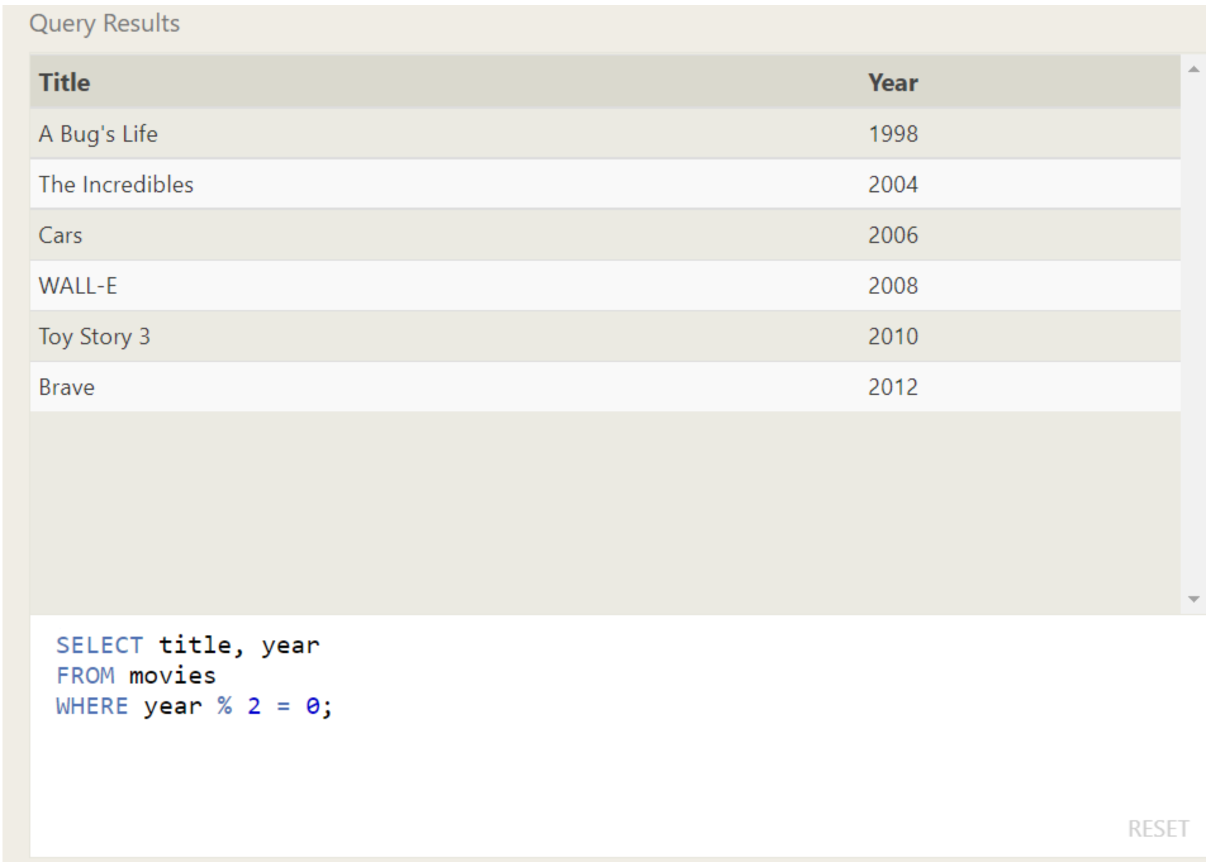


Title	Rating_percent
Finding Nemo	82
Monsters University	74
Ratatouille	80
Cars 2	64
Toy Story 2	79
The Incredibles	80
WALL-E	85
Toy Story 3	84
Toy Story	83
Cars	72

```
SELECT title, rating * 10 AS rating_percent
FROM movies
  JOIN boxoffice
    ON movies.id = boxoffice.movie_id;
```

9(iii). List all movies that were released on even number years

```
SELECT title, year
FROM movies
WHERE year % 2 = 0;
```


Query Results

Title	Year
A Bug's Life	1998
The Incredibles	2004
Cars	2006
WALL-E	2008
Toy Story 3	2010
Brave	2012

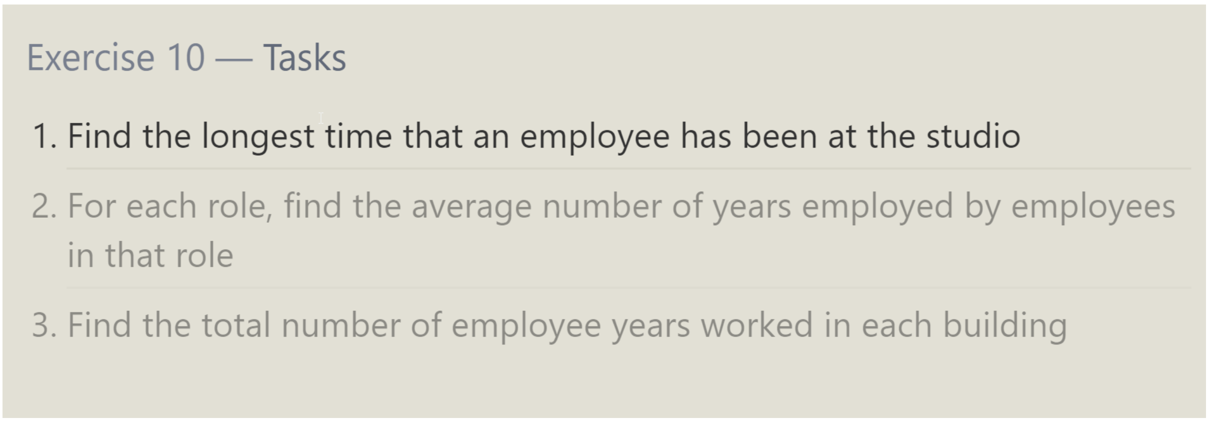
```
SELECT title, year
FROM movies
WHERE year % 2 = 0;
```

Exercise 10

SQLBolt - Learn SQL - SQL Lesson 10: Queries with aggregates (Pt. 1)

In addition to the simple expressions that we introduced last lesson, SQL also supports the use of aggregate expressions (or functions) that allow you to summarize information about a group of rows of data.

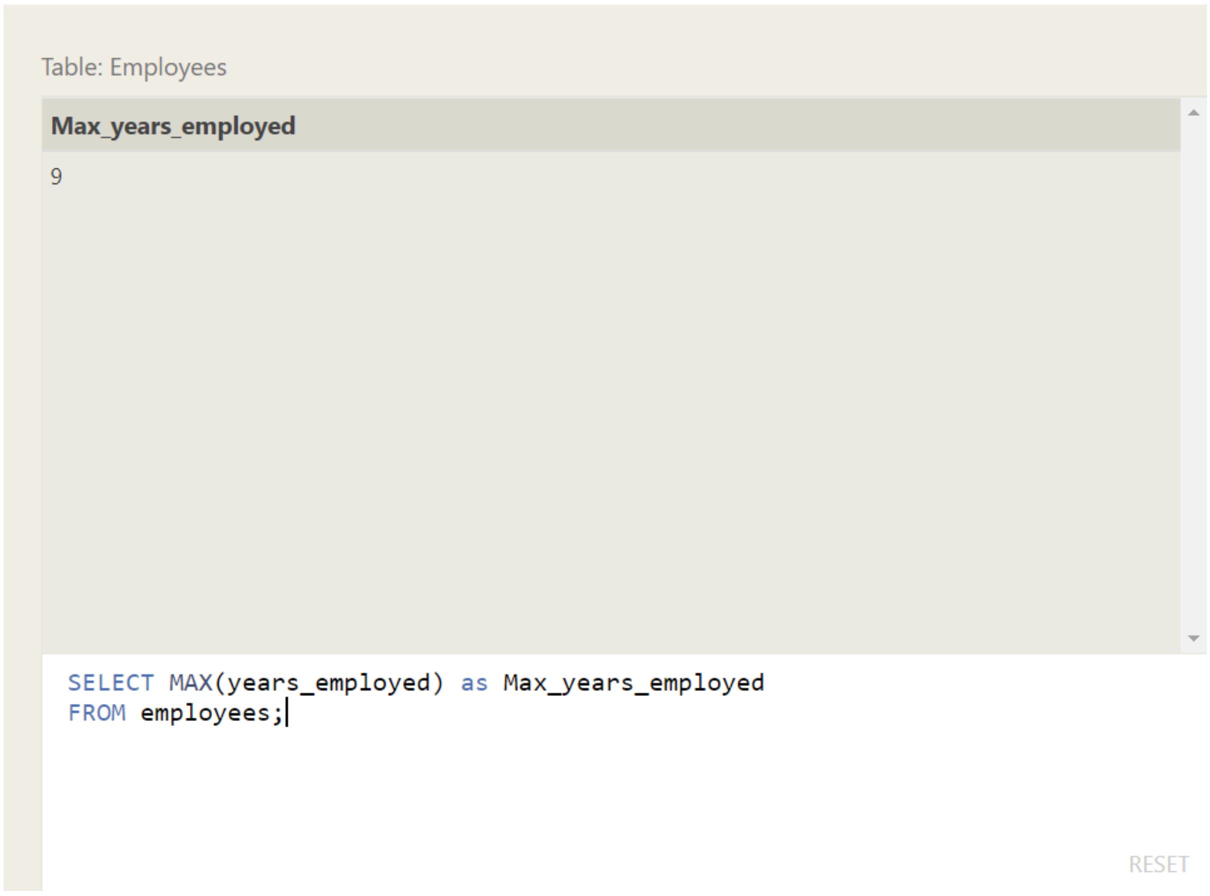
https://sqlbolt.com/lesson/select_queries_with_aggregates


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

10(i). Find the longest time that an employee has been at the studio

```
SELECT MAX(years_employed) as Max_years_employed
FROM employees;
```



Max_years_employed
9

```
SELECT MAX(years_employed) as Max_years_employed
FROM employees;
```

10(ii).For each role, find the average number of years employed by employees in that role

```
SELECT role, AVG(years_employed) as Average_years_employed
FROM employees
GROUP BY role;
```

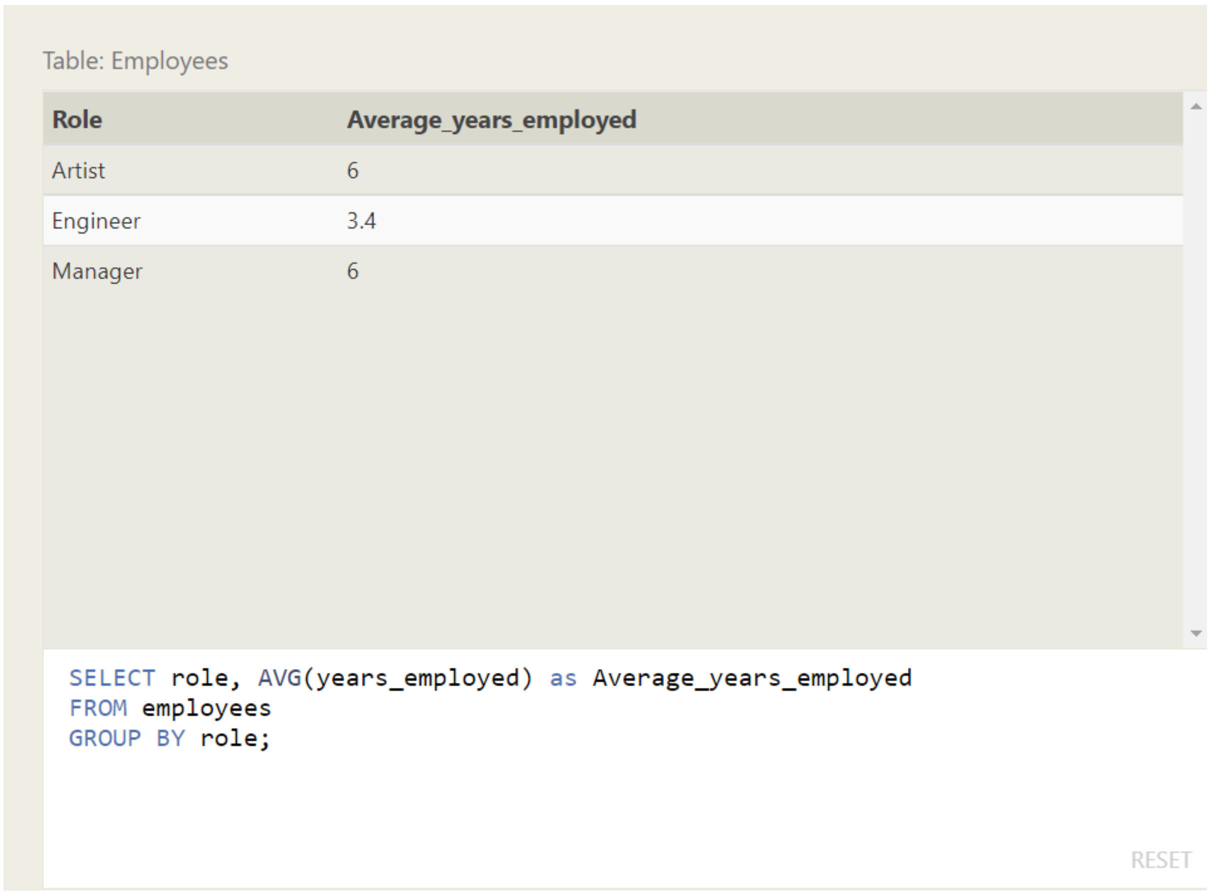


Table: Employees

Role	Average_years_employed
Artist	6
Engineer	3.4
Manager	6

```sql
SELECT role, AVG(years_employed) as Average_years_employed
FROM employees
GROUP BY role;
```

10(iii). Find the total number of employee years worked in each building

```sql
SELECT building, SUM(years_employed) as Total_years
FROM employees
GROUP BY building;
```

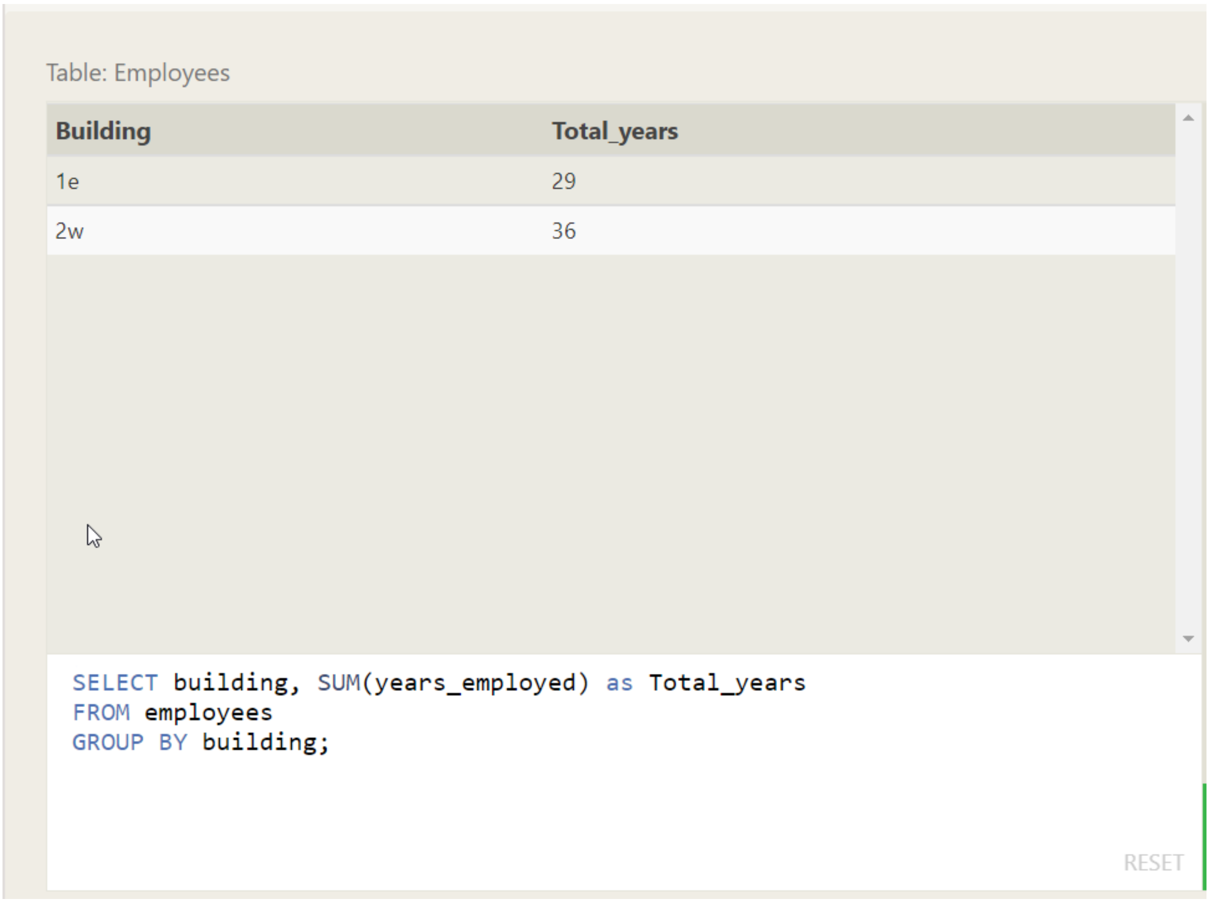


Table: Employees

Building	Total_years
1e	29
2w	36

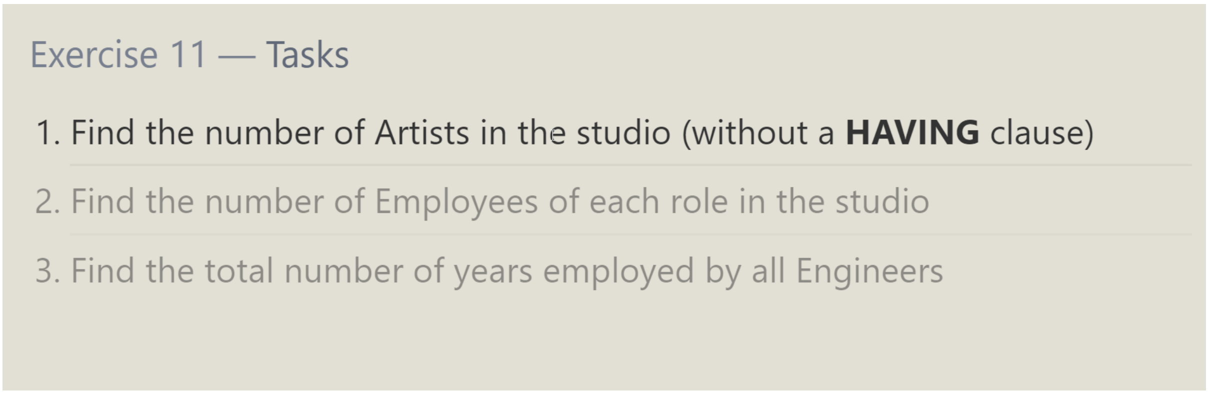
```
SELECT building, SUM(years_employed) as Total_years
FROM employees
GROUP BY building;
```

Exercise 11

SQLBolt - Learn SQL - SQL Lesson 11: Queries with aggregates (Pt. 2)

Our queries are getting fairly complex, but we have nearly introduced all the important parts of a SELECT query. One thing that you might have noticed is that if the GROUP BY clause is executed after the WHERE clause (which filters the rows which are to be grouped), then how exactly do we filter the grouped rows?

https://sqlbolt.com/lesson/select_queries_with_aggregates_pt_2


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

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

```
SELECT role, COUNT(*) as No_of_artists
FROM employees
WHERE role = "Artist";
```

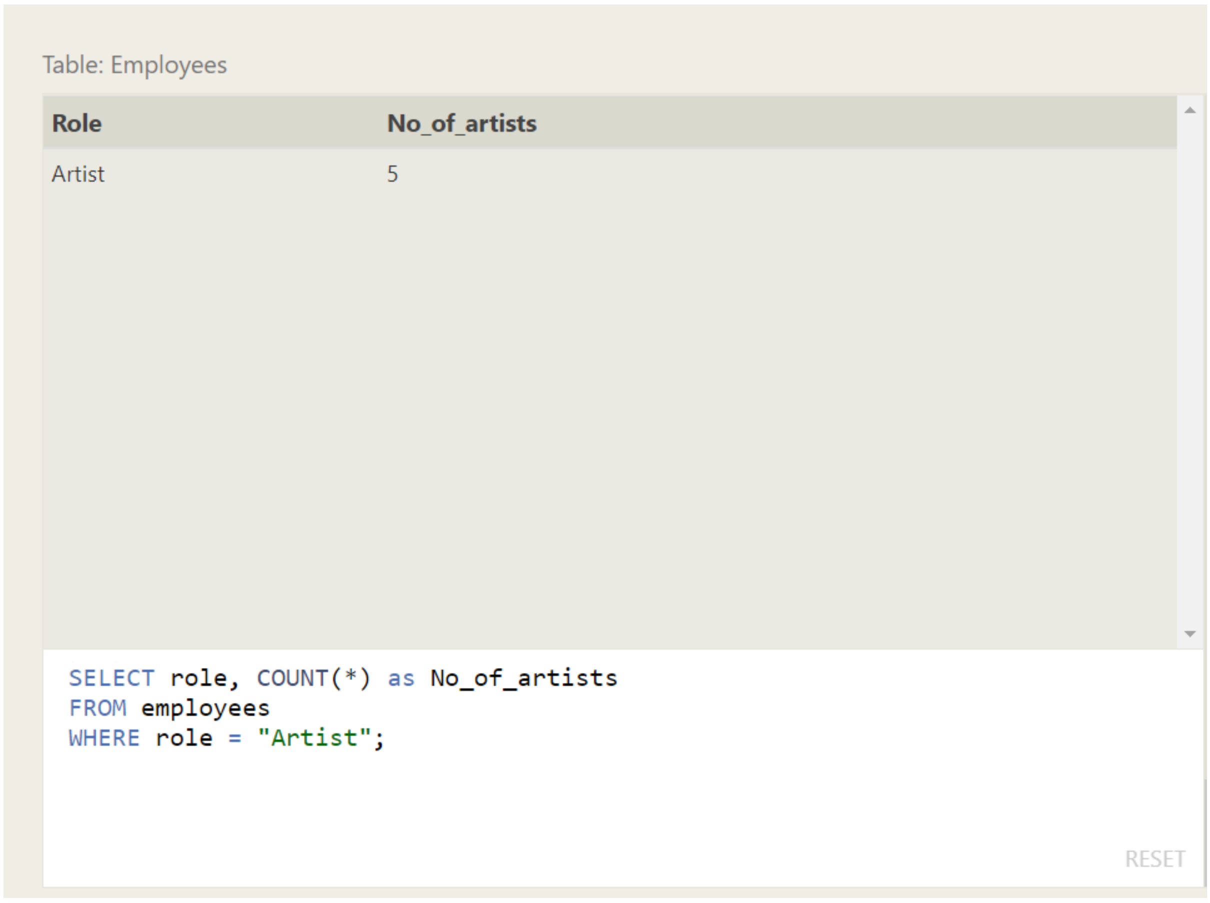


Table: Employees

Role	No_of_artists
Artist	5

```
SELECT role, COUNT(*) as No_of_artists
FROM employees
WHERE role = "Artist";
```

11(ii). Find the number of Employees of each role in the studio

```
SELECT role, COUNT(*)as No_of_Employees
FROM employees
GROUP BY role;
```

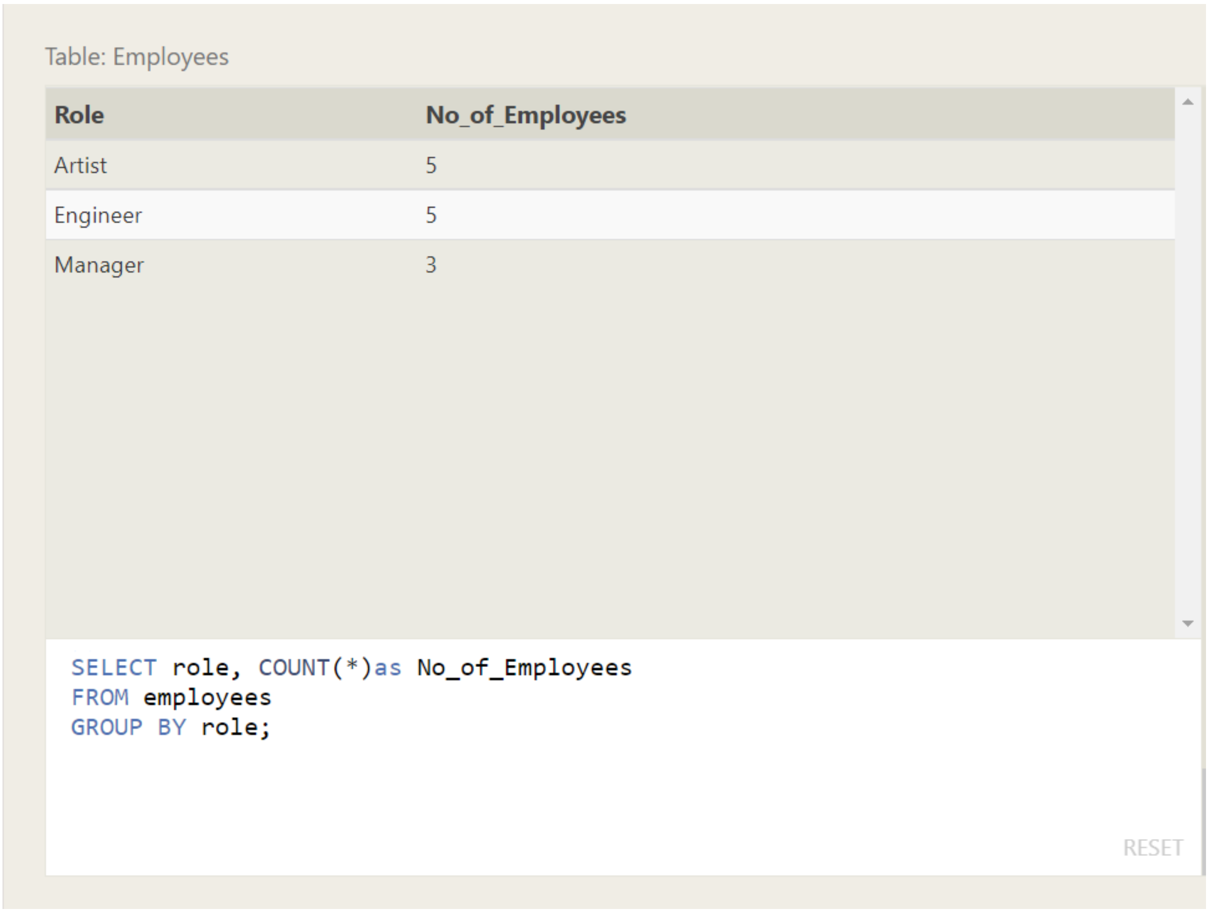


Table: Employees

Role	No_of_Employees
Artist	5
Engineer	5
Manager	3

```
SELECT role, COUNT(*)as No_of_Employees
FROM employees
GROUP BY role;
```

11(iii). Find the total number of years employed by all Engineers

```
SELECT role, SUM(years_employed)as Total_Years_Employed
FROM employees
GROUP BY role
HAVING role = "Engineer";
```

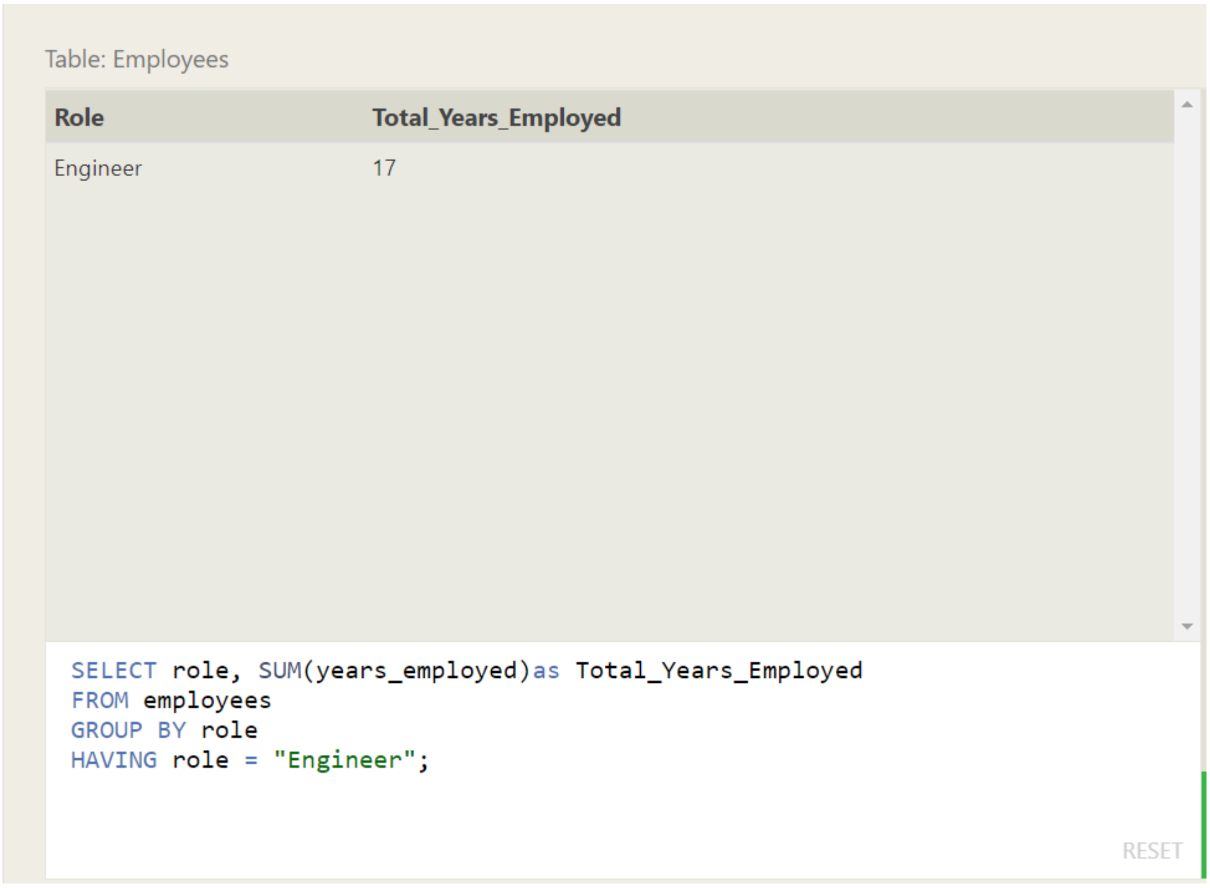


Table: Employees

Role	Total_Years_Employed
Engineer	17

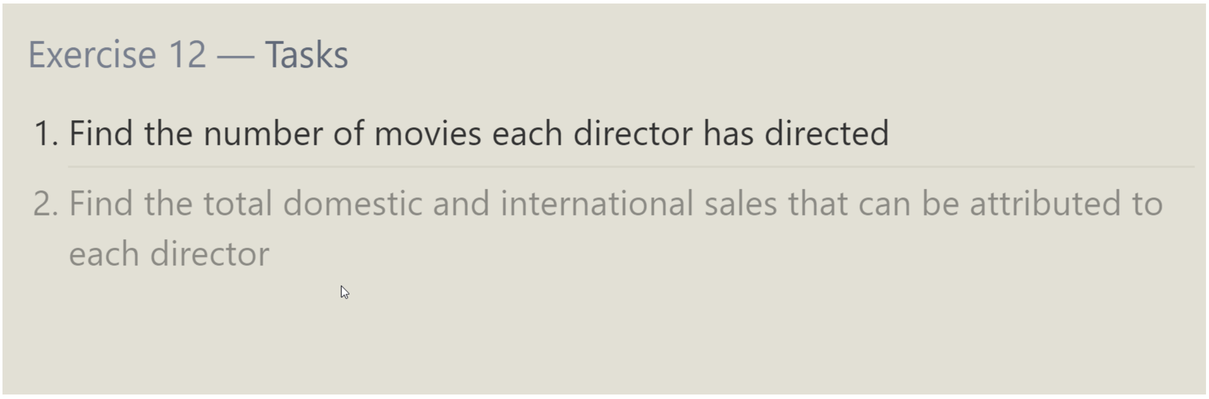
```
SELECT role, SUM(years_employed)as Total_Years_Employed
FROM employees
GROUP BY role
HAVING role = "Engineer";
```

Exercise 12

SQLBolt - Learn SQL - SQL Lesson 12: Order of execution of a Query

Now that we have an idea of all the parts of a query, we can now talk about how they all fit together in the context of a complete query. SELECT DISTINCT column, AGG_FUNC(column_or_expression), ...

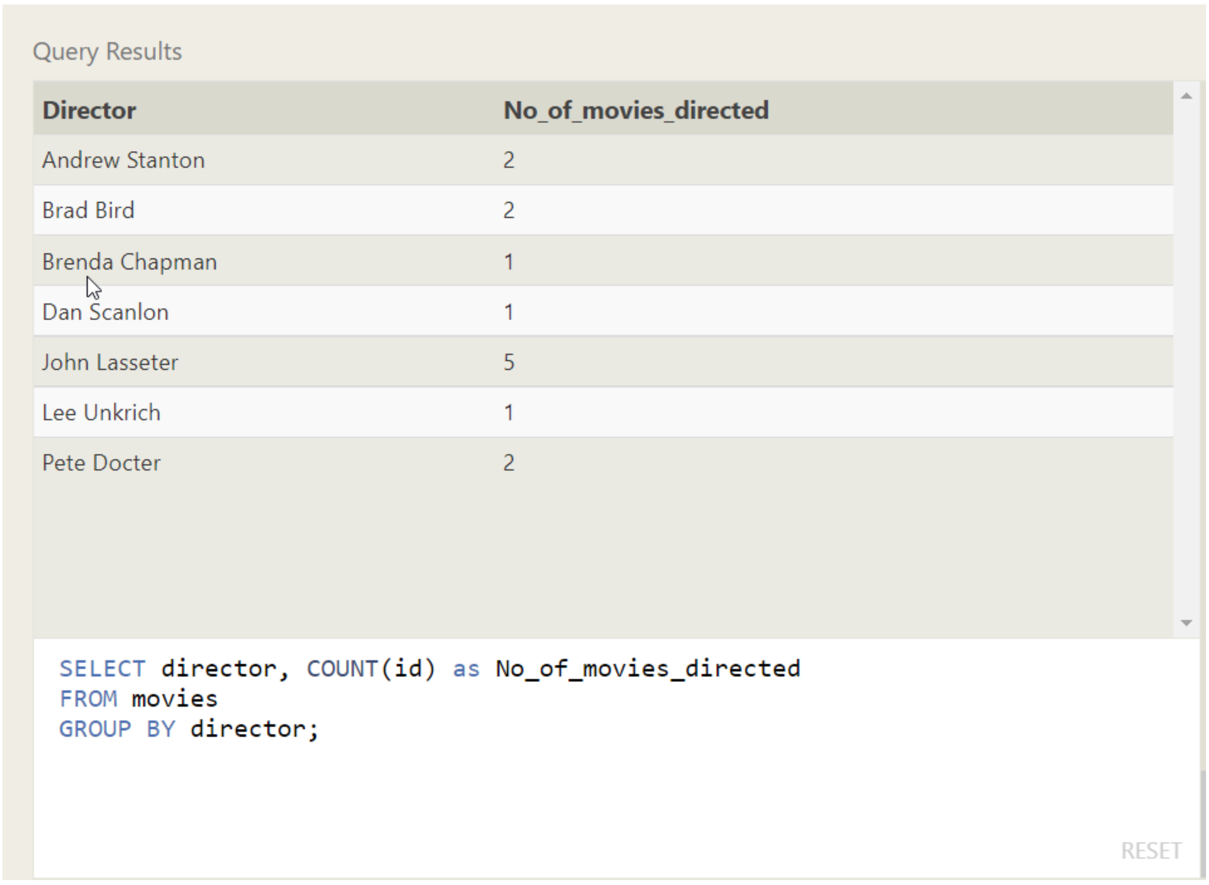
https://sqlbolt.com/lesson/select_queries_order_of_execution


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

12(i).Find the number of movies each director has directed

```
SELECT director, COUNT(id) as No_of_movies_directed
FROM movies
GROUP BY director;
```



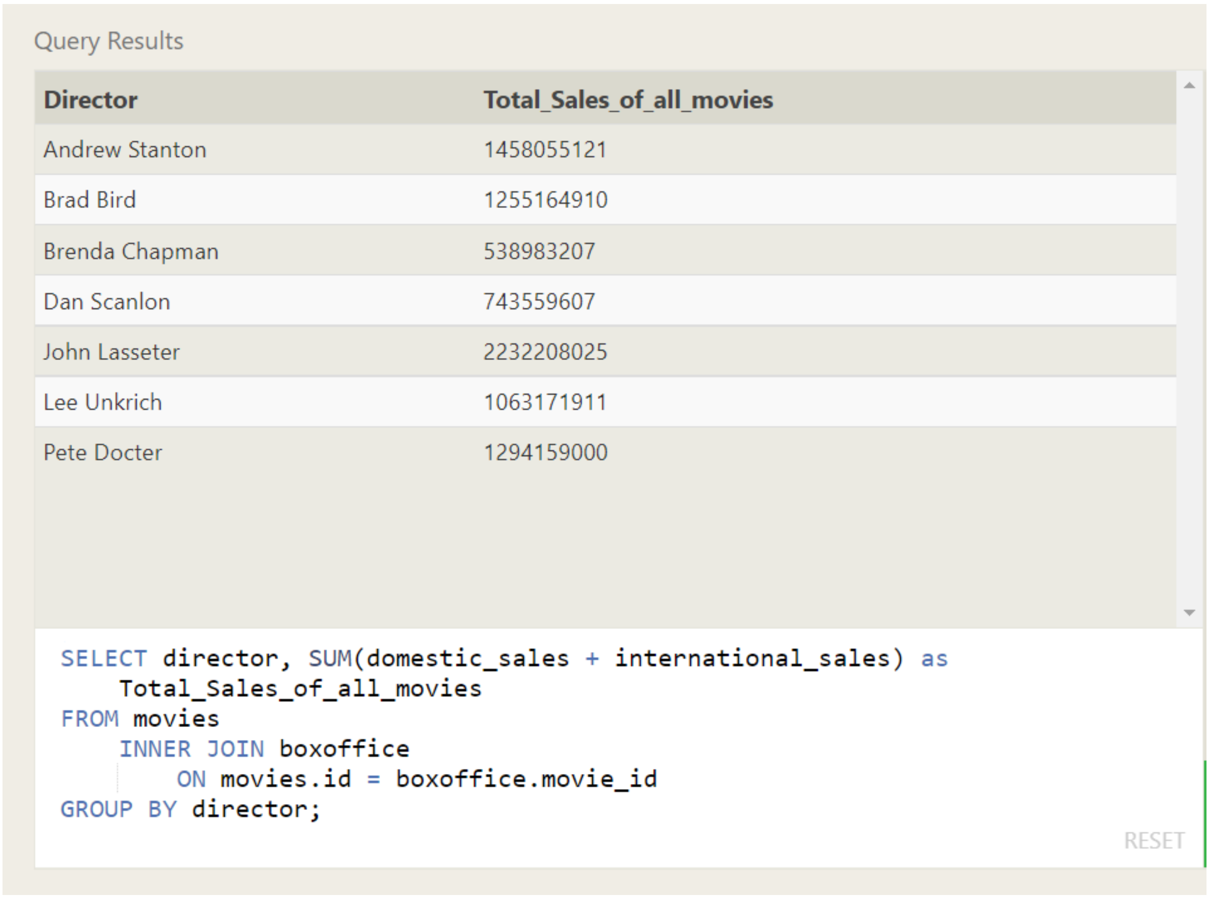
Query Results

Director	No_of_movies_directed
Andrew Stanton	2
Brad Bird	2
Brenda Chapman	1
Dan Scanlon	1
John Lasseter	5
Lee Unkrich	1
Pete Docter	2

```
SELECT director, COUNT(id) as No_of_movies_directed
FROM movies
GROUP BY director;
```

12(ii). Find the total domestic and international sales that can be attributed to each director

```
SELECT director, SUM(domestic_sales + international_sales) as Total_Sales_of_all_movies
FROM movies
    INNER JOIN boxoffice
        ON movies.id = boxoffice.movie_id
GROUP BY director;
```



Director	Total_Sales_of_all_movies
Andrew Stanton	1458055121
Brad Bird	1255164910
Brenda Chapman	538983207
Dan Scanlon	743559607
John Lasseter	2232208025
Lee Unkrich	1063171911
Pete Docter	1294159000

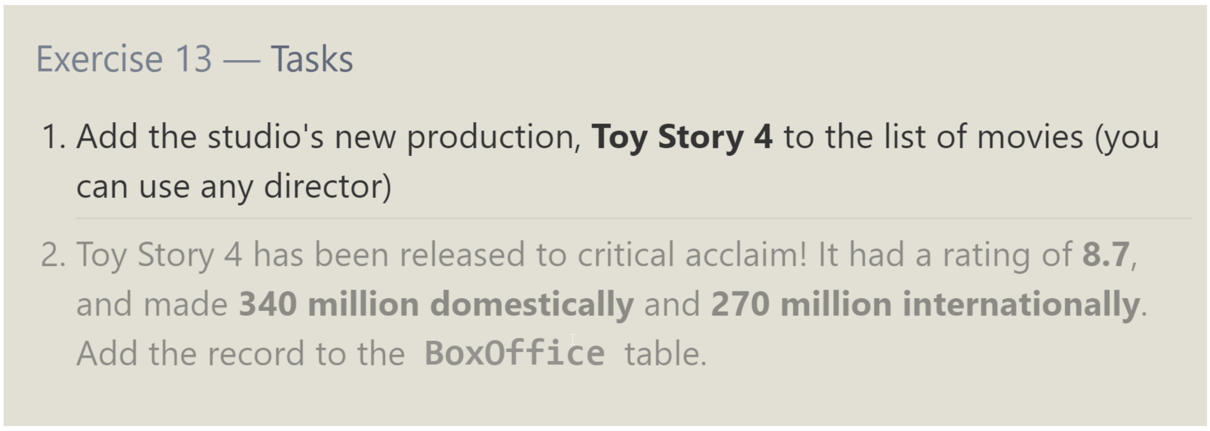
```
SELECT director, SUM(domestic_sales + international_sales) as
    Total_Sales_of_all_movies
FROM movies
    INNER JOIN boxoffice
        ON movies.id = boxoffice.movie_id
GROUP BY director;
```

Exercise 13

SQLBolt - Learn SQL - SQL Lesson 13: Inserting rows

We've spent quite a few lessons on how to query for data in a database, so it's time to start learning a bit about SQL schemas and how to add new data. We previously described a table in a database as a two-dimensional set of rows and columns, with the columns being the properties and the rows being instances

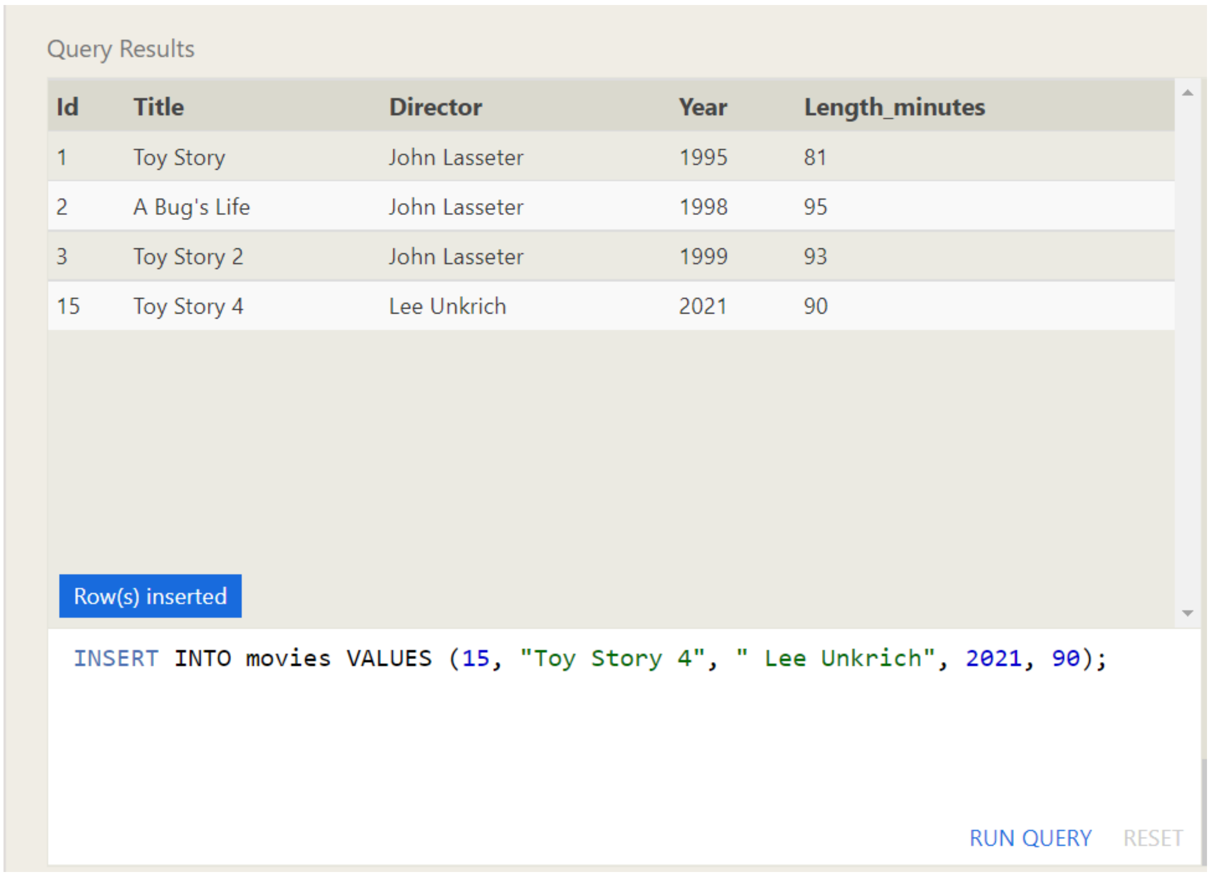
https://sqlbolt.com/lesson/inserting_rows


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.

13(i). 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", " Lee Unkrich", 2021, 90);
```


Query Results

Id	Title	Director	Year	Length_minutes
1	Toy Story	John Lasseter	1995	81
2	A Bug's Life	John Lasseter	1998	95
3	Toy Story 2	John Lasseter	1999	93
15	Toy Story 4	Lee Unkrich	2021	90

Row(s) inserted

```
INSERT INTO movies VALUES (15, "Toy Story 4", " Lee Unkrich", 2021, 90);
```

13(ii). 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);
```

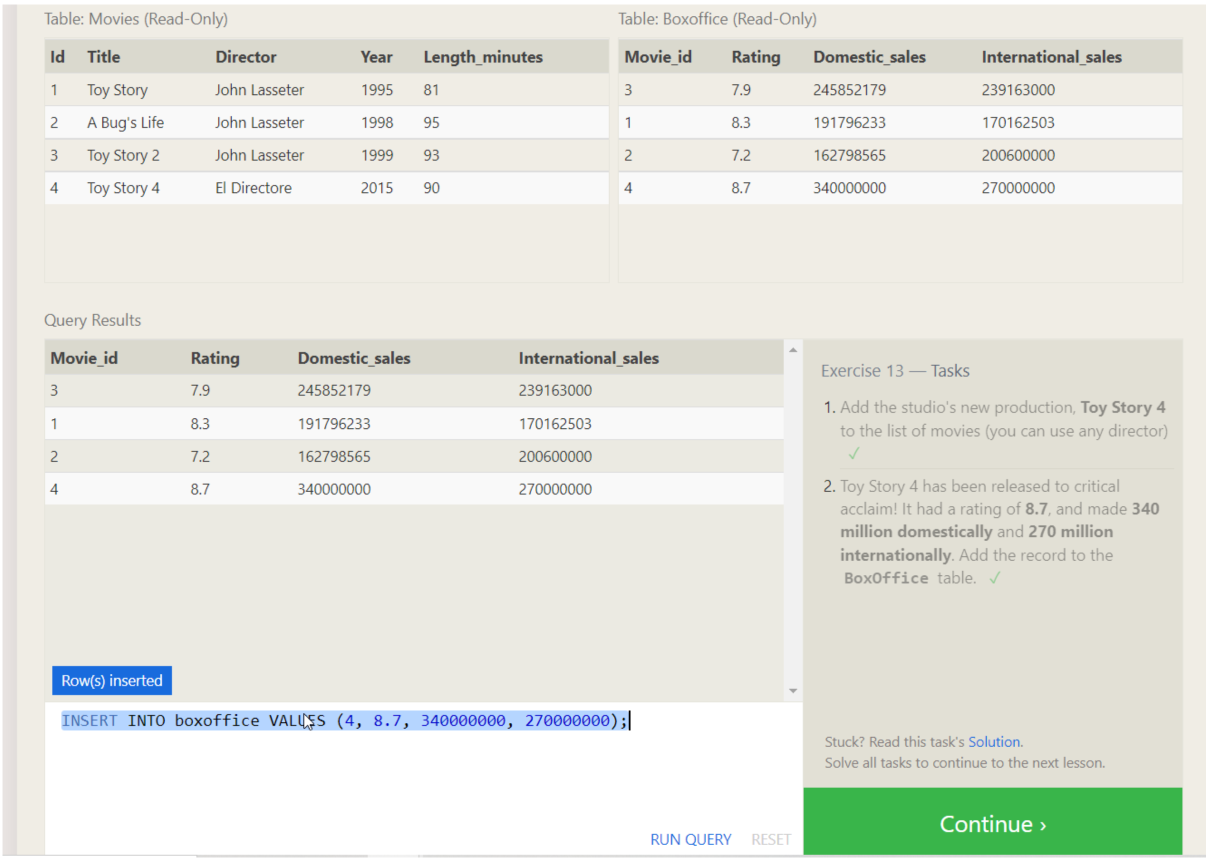


Table: Movies (Read-Only)

Id	Title	Director	Year	Length_minutes
1	Toy Story	John Lasseter	1995	81
2	A Bug's Life	John Lasseter	1998	95
3	Toy Story 2	John Lasseter	1999	93
4	Toy Story 4	El Directore	2015	90

Table: Boxoffice (Read-Only)

Movie_id	Rating	Domestic_sales	International_sales
3	7.9	245852179	239163000
1	8.3	191796233	170162503
2	7.2	162798565	200600000
4	8.7	340000000	270000000

Query Results

Movie_id	Rating	Domestic_sales	International_sales
3	7.9	245852179	239163000
1	8.3	191796233	170162503
2	7.2	162798565	200600000
4	8.7	340000000	270000000

Row(s) inserted

```
INSERT INTO boxoffice VALUES (4, 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. ✓

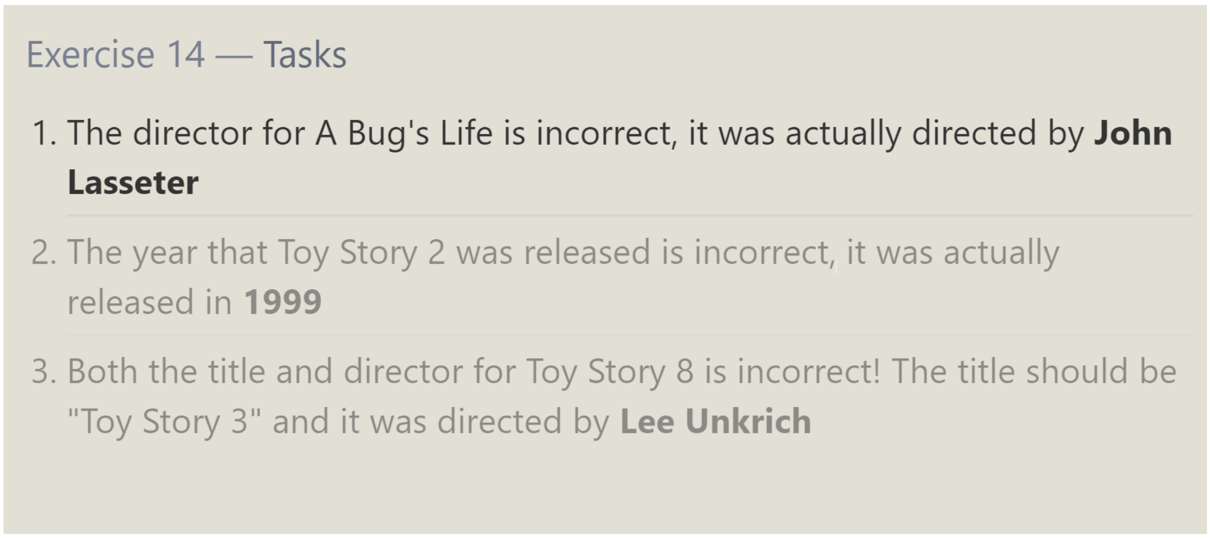
Stuck? Read this task's Solution.
Solve all tasks to continue to the next lesson.

Exercise 14

SQLBolt - Learn SQL - SQL Lesson 14: Updating rows

In addition to adding new data, a common task is to update existing data, which can be done using an UPDATE statement. Similar to the INSERT statement, you have to specify exactly which table, columns, and rows to update. In addition, the data you are updating has to match the data type of the columns in the

https://sqlbolt.com/lesson/updating_rows


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**

14(i). The director for A Bug's Life is incorrect, it was actually directed by John Lasseter

```
UPDATE movies
SET director = "John Lasseter"
WHERE id = 2;
```

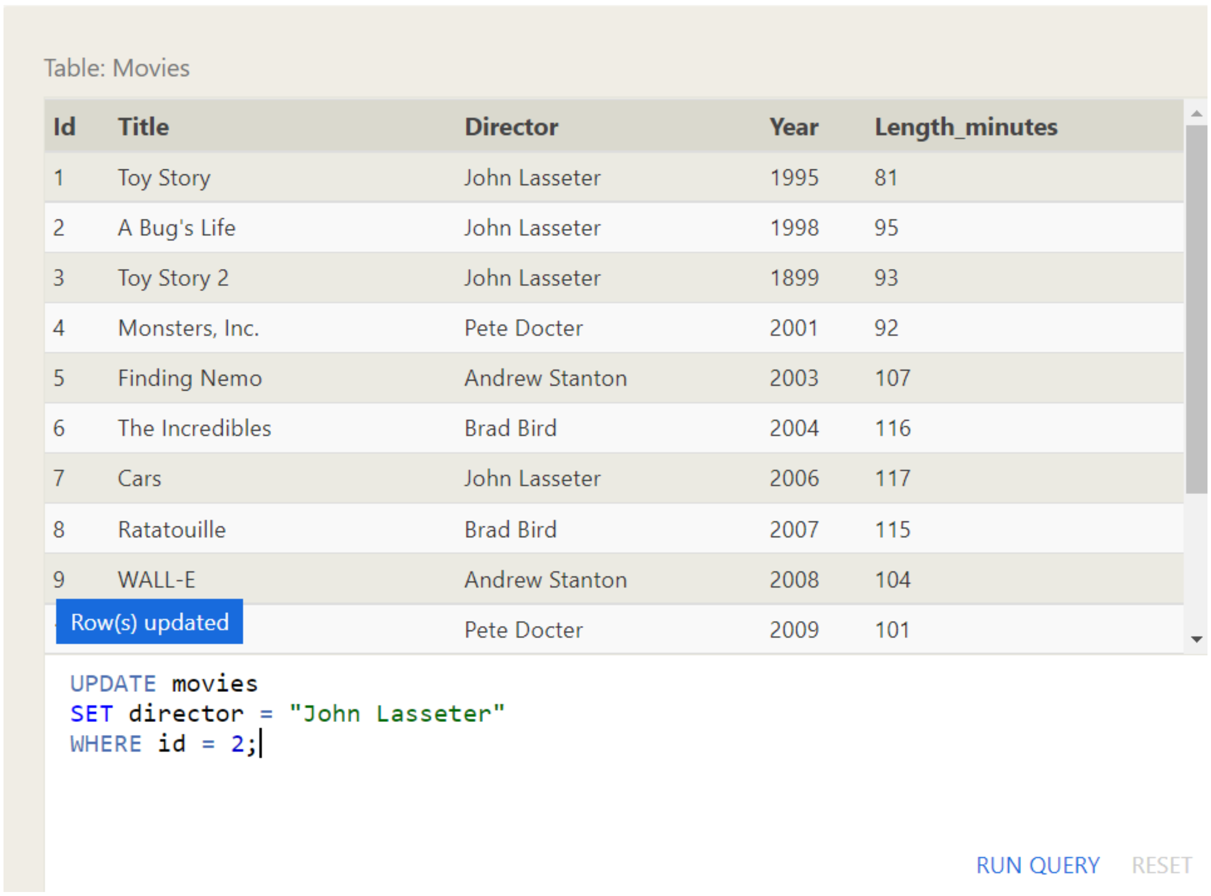


Table: Movies

Id	Title	Director	Year	Length_minutes
1	Toy Story	John Lasseter	1995	81
2	A Bug's Life	John Lasseter	1998	95
3	Toy Story 2	John Lasseter	1899	93
4	Monsters, Inc.	Pete Docter	2001	92
5	Finding Nemo	Andrew Stanton	2003	107
6	The Incredibles	Brad Bird	2004	116
7	Cars	John Lasseter	2006	117
8	Ratatouille	Brad Bird	2007	115
9	WALL-E	Andrew Stanton	2008	104
		Pete Docter	2009	101

```
UPDATE movies
SET director = "John Lasseter"
WHERE id = 2;
```

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

```
UPDATE movies
SET year = 1999
WHERE id = 3;
```

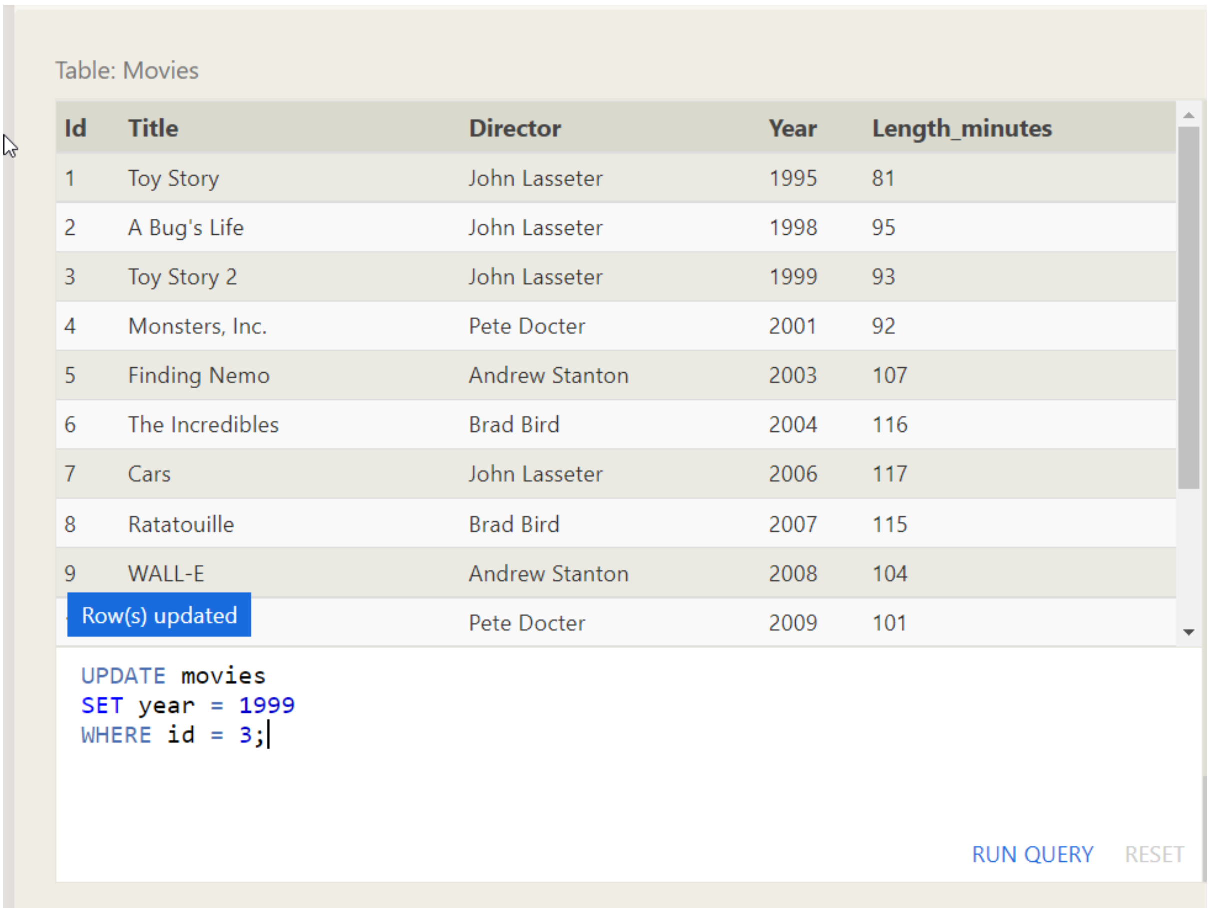


Table: Movies

Id	Title	Director	Year	Length_minutes
1	Toy Story	John Lasseter	1995	81
2	A Bug's Life	John Lasseter	1998	95
3	Toy Story 2	John Lasseter	1999	93
4	Monsters, Inc.	Pete Docter	2001	92
5	Finding Nemo	Andrew Stanton	2003	107
6	The Incredibles	Brad Bird	2004	116
7	Cars	John Lasseter	2006	117
8	Ratatouille	Brad Bird	2007	115
9	WALL-E	Andrew Stanton	2008	104
	Row(s) updated	Pete Docter	2009	101

```
UPDATE movies
SET year = 1999
WHERE id = 3;
```

14(iii). 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 id = 11;
```

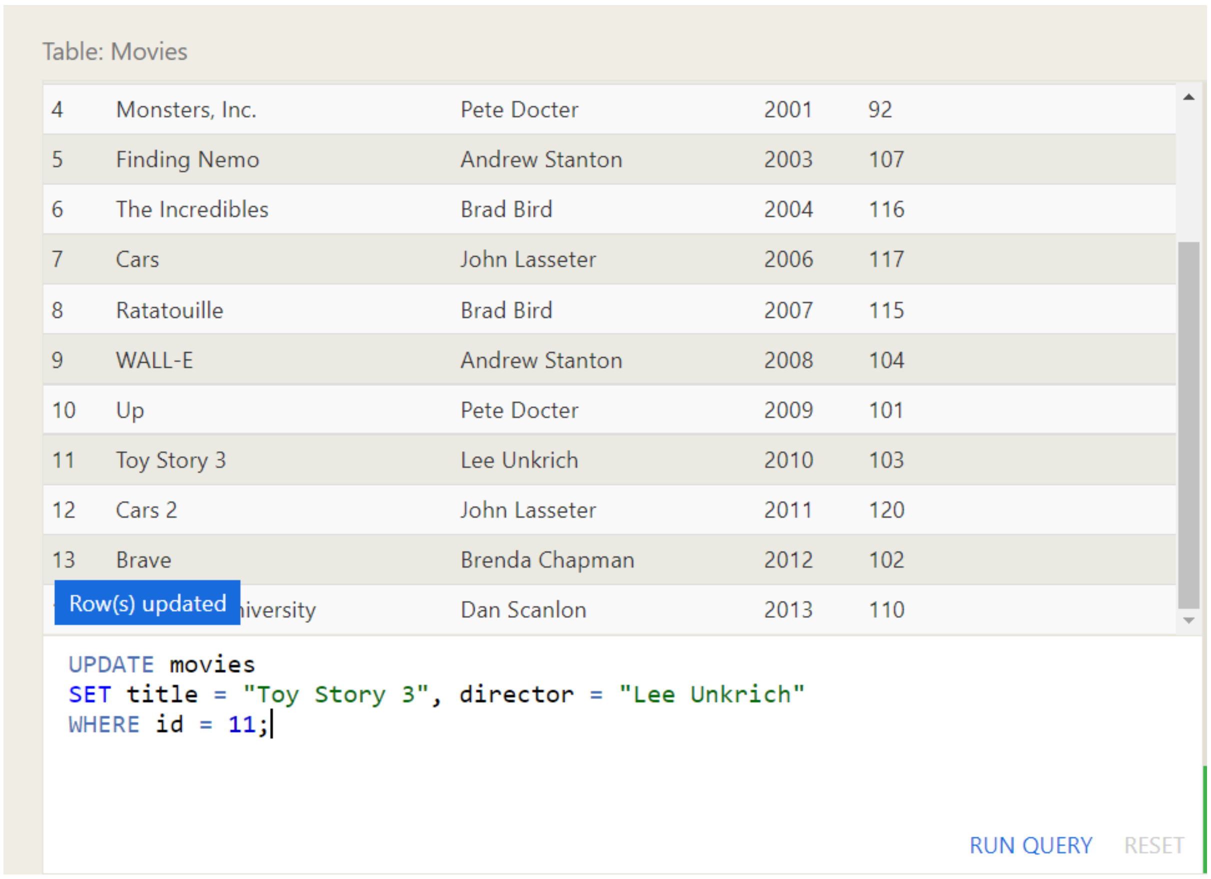


Table: Movies

4	Monsters, Inc.	Pete Docter	2001	92
5	Finding Nemo	Andrew Stanton	2003	107
6	The Incredibles	Brad Bird	2004	116
7	Cars	John Lasseter	2006	117
8	Ratatouille	Brad Bird	2007	115
9	WALL-E	Andrew Stanton	2008	104
10	Up	Pete Docter	2009	101
11	Toy Story 3	Lee Unkrich	2010	103
12	Cars 2	John Lasseter	2011	120
13	Brave	Brenda Chapman	2012	102
	iversity	Dan Scanlon	2013	110

Row(s) updated

```
UPDATE movies
SET title = "Toy Story 3", director = "Lee Unkrich"
WHERE id = 11;
```

Exercise 15

SQLBolt - Learn SQL - SQL Lesson 15: Deleting rows

When you need to delete data from a table in the database, you can use a DELETE statement, which describes the table to act on, and the rows of the table to delete through the WHERE clause.

https://sqlbolt.com/lesson/deleting_rows

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.

15(i). This database is getting too big, lets remove all movies that were released before 2005.

```
DELETE FROM movies
where year < 2005;
```

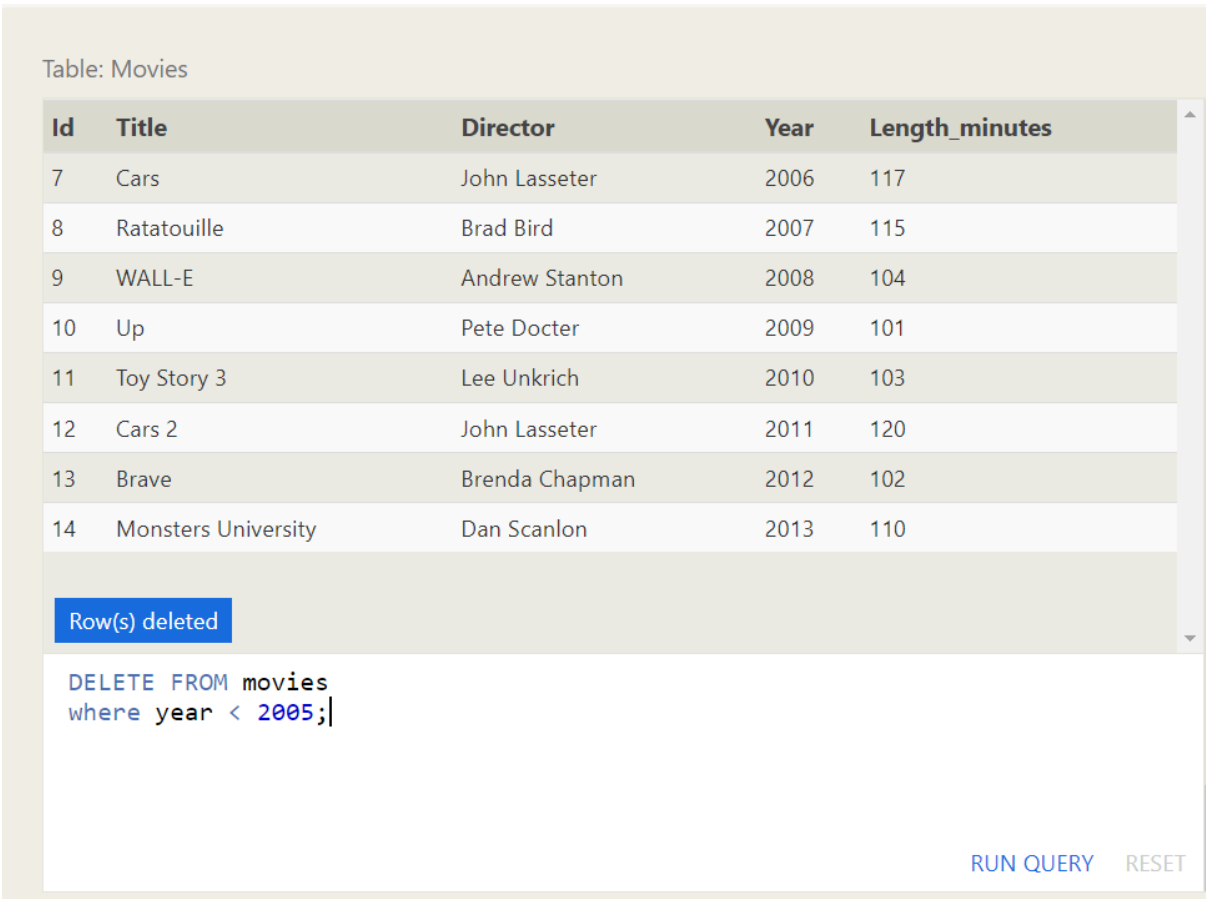


Table: Movies

Id	Title	Director	Year	Length_minutes
7	Cars	John Lasseter	2006	117
8	Ratatouille	Brad Bird	2007	115
9	WALL-E	Andrew Stanton	2008	104
10	Up	Pete Docter	2009	101
11	Toy Story 3	Lee Unkrich	2010	103
12	Cars 2	John Lasseter	2011	120
13	Brave	Brenda Chapman	2012	102
14	Monsters University	Dan Scanlon	2013	110

Row(s) deleted

```
DELETE FROM movies
where year < 2005;
```

15(ii). Andrew Stanton has also left the studio, so please remove all movies directed by him.

```
DELETE FROM movies
where director = "Andrew Stanton";
```

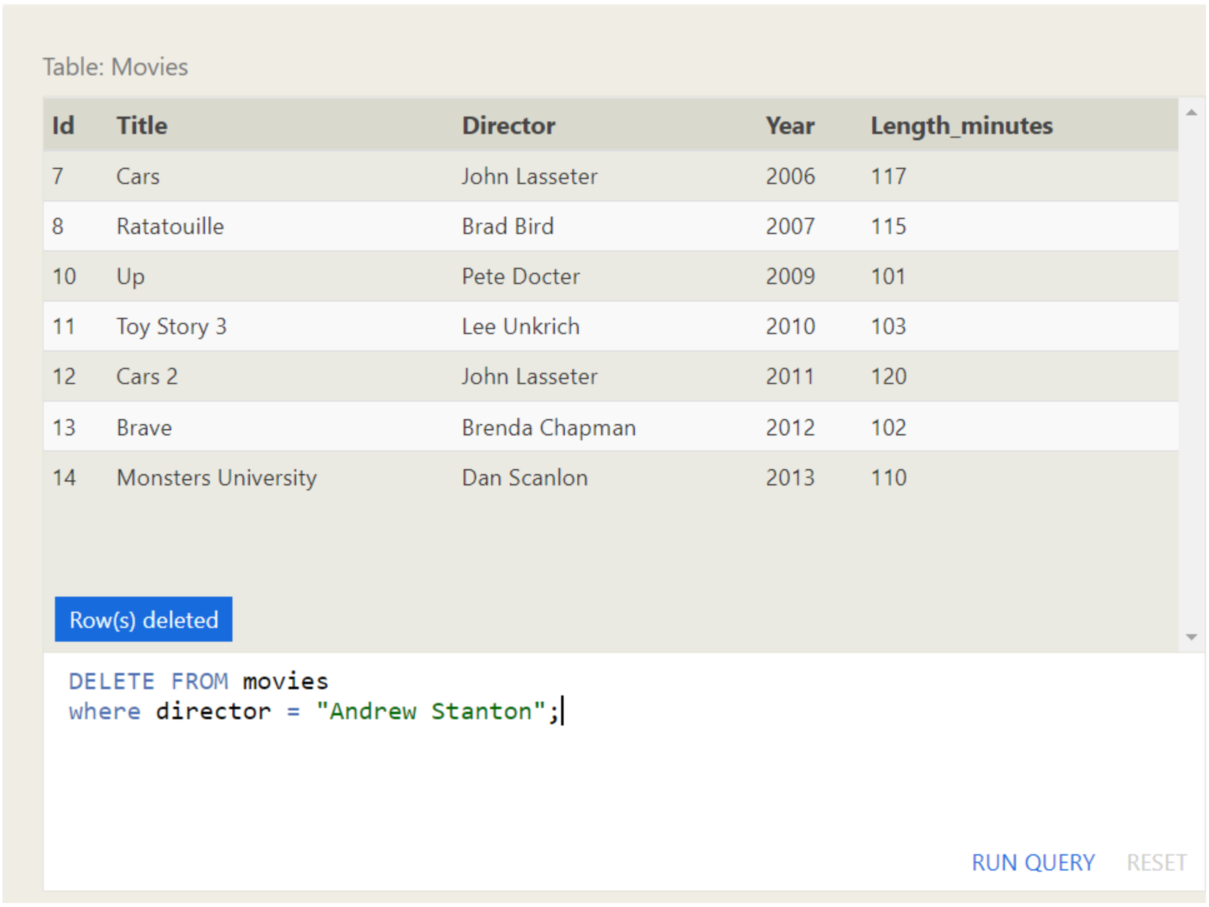


Table: Movies

Id	Title	Director	Year	Length_minutes
7	Cars	John Lasseter	2006	117
8	Ratatouille	Brad Bird	2007	115
10	Up	Pete Docter	2009	101
11	Toy Story 3	Lee Unkrich	2010	103
12	Cars 2	John Lasseter	2011	120
13	Brave	Brenda Chapman	2012	102
14	Monsters University	Dan Scanlon	2013	110

Row(s) deleted

```
DELETE FROM movies
where director = "Andrew Stanton";
```

Exercise 16

SQLBolt - Learn SQL - SQL Lesson 16: Creating tables

When you have new entities and relationships to store in your database, you can create a new database table using the CREATE TABLE statement. Create table statement w/ optional table constraint and default value CREATE TABLE IF NOT EXISTS mytable (column DataType TableConstraint DEFAULT

https://sqlbolt.com/lesson/creating_tables

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 FLOAT,
    Download_count INTEGER
);
```

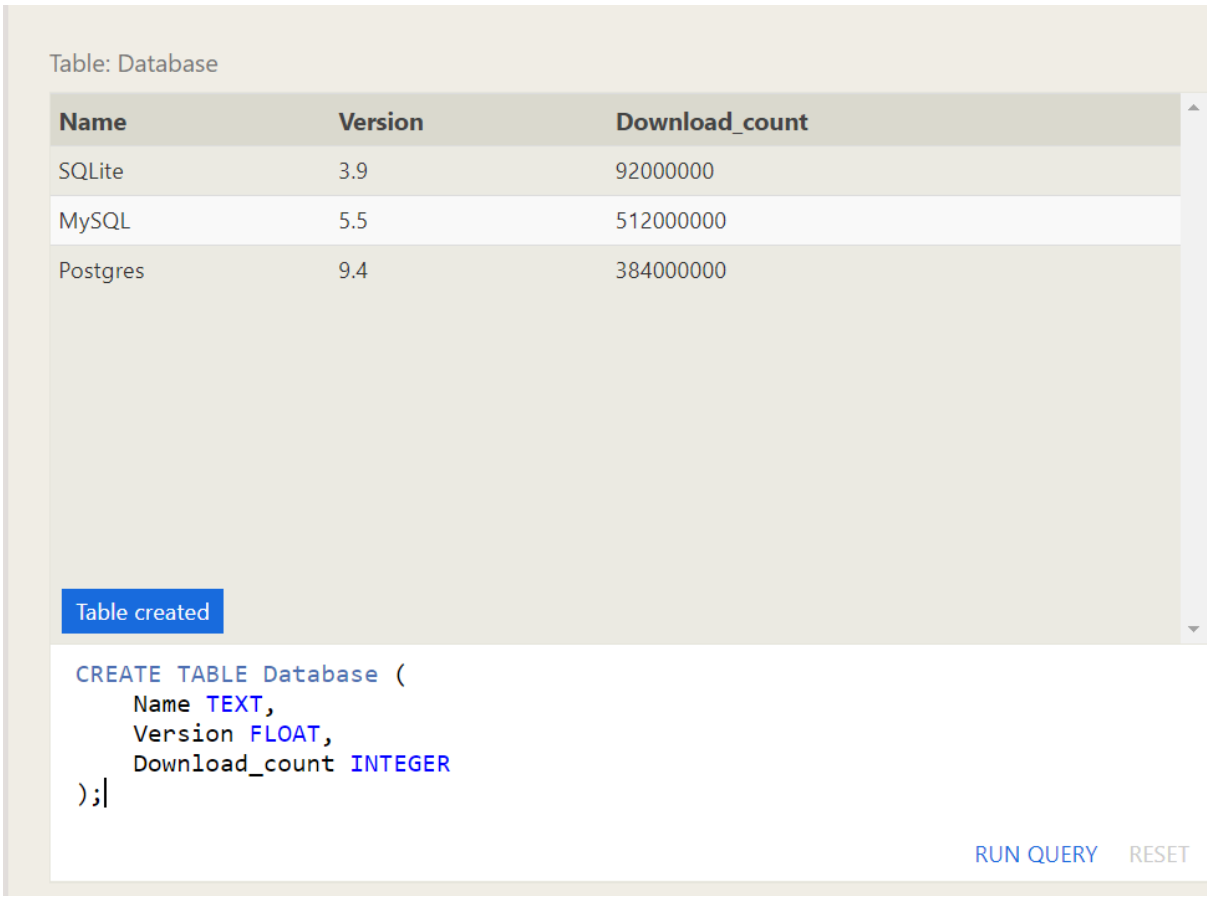


Table: Database

Name	Version	Download_count
SQLite	3.9	92000000
MySQL	5.5	512000000
Postgres	9.4	384000000

Table created

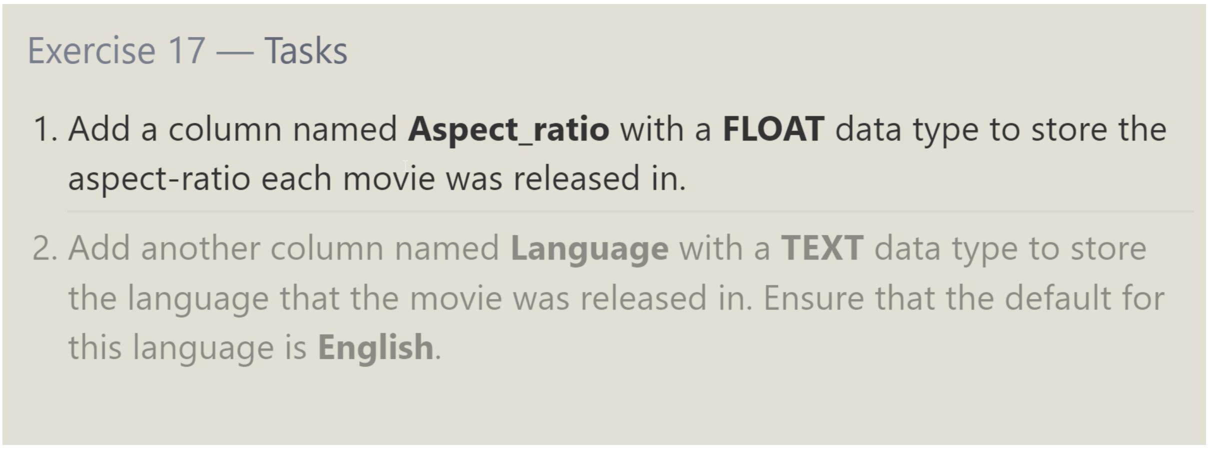
```
CREATE TABLE Database (
    Name TEXT,
    Version FLOAT,
    Download_count INTEGER
);
```

Exercise 17

SQLBolt - Learn SQL - SQL Lesson 17: Altering tables

As your data changes over time, SQL provides a way for you to update your corresponding tables and database schemas by using the ALTER TABLE statement to add, remove, or modify columns and table constraints. The syntax for adding a new column is similar to the syntax when creating new rows in the

https://sqlbolt.com/lesson/altering_tables



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**.

17(i). 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 DEFAULT 2.39;
```

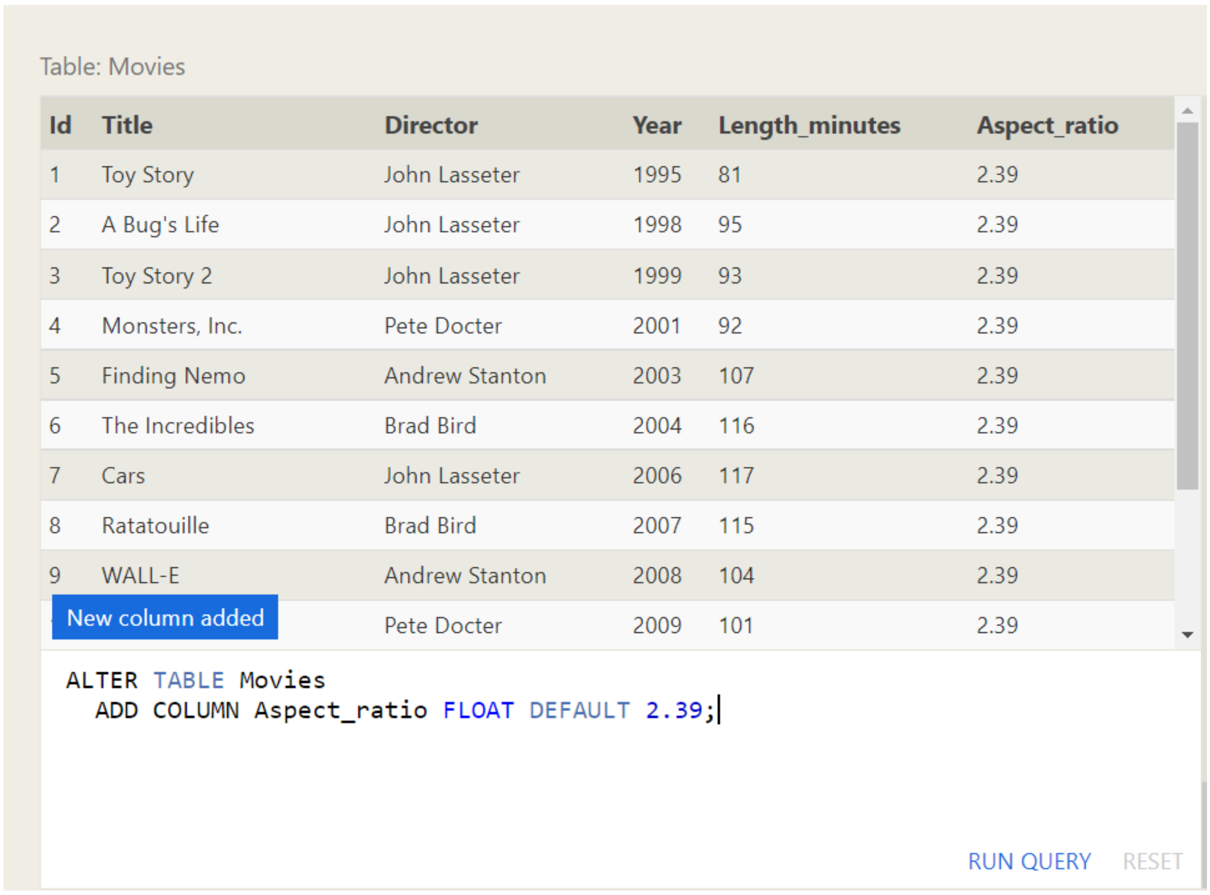


Table: Movies

Id	Title	Director	Year	Length_minutes	Aspect_ratio
1	Toy Story	John Lasseter	1995	81	2.39
2	A Bug's Life	John Lasseter	1998	95	2.39
3	Toy Story 2	John Lasseter	1999	93	2.39
4	Monsters, Inc.	Pete Docter	2001	92	2.39
5	Finding Nemo	Andrew Stanton	2003	107	2.39
6	The Incredibles	Brad Bird	2004	116	2.39
7	Cars	John Lasseter	2006	117	2.39
8	Ratatouille	Brad Bird	2007	115	2.39
9	WALL-E	Andrew Stanton	2008	104	2.39
[illegible]	[illegible]	Pete Docter	2009	101	2.39

New column added

```
ALTER TABLE Movies
  ADD COLUMN Aspect_ratio FLOAT DEFAULT 2.39;
```

17(ii). 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";
```

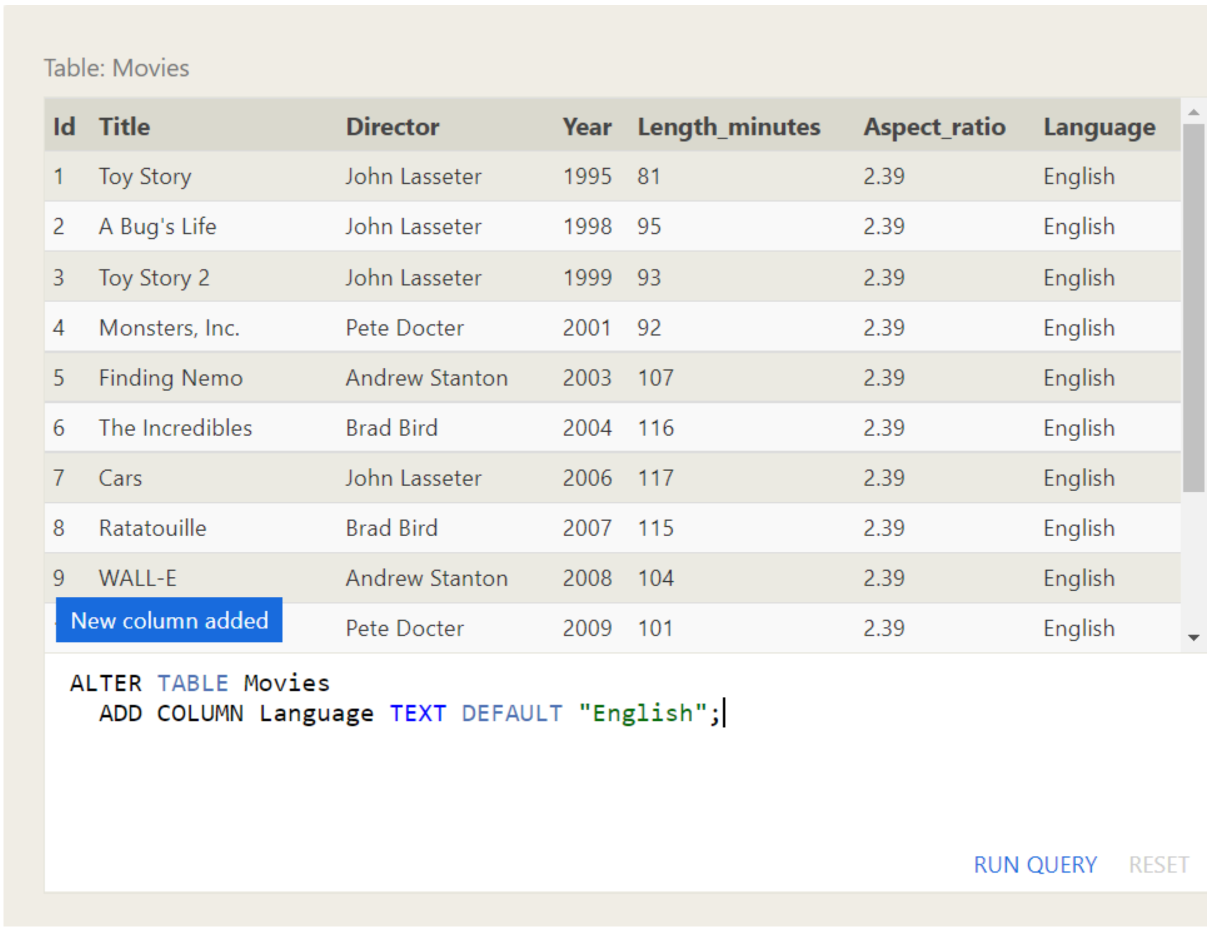


Table: Movies

Id	Title	Director	Year	Length_minutes	Aspect_ratio	Language
1	Toy Story	John Lasseter	1995	81	2.39	English
2	A Bug's Life	John Lasseter	1998	95	2.39	English
3	Toy Story 2	John Lasseter	1999	93	2.39	English
4	Monsters, Inc.	Pete Docter	2001	92	2.39	English
5	Finding Nemo	Andrew Stanton	2003	107	2.39	English
6	The Incredibles	Brad Bird	2004	116	2.39	English
7	Cars	John Lasseter	2006	117	2.39	English
8	Ratatouille	Brad Bird	2007	115	2.39	English
9	WALL-E	Andrew Stanton	2008	104	2.39	English
		Pete Docter	2009	101	2.39	English

New column added

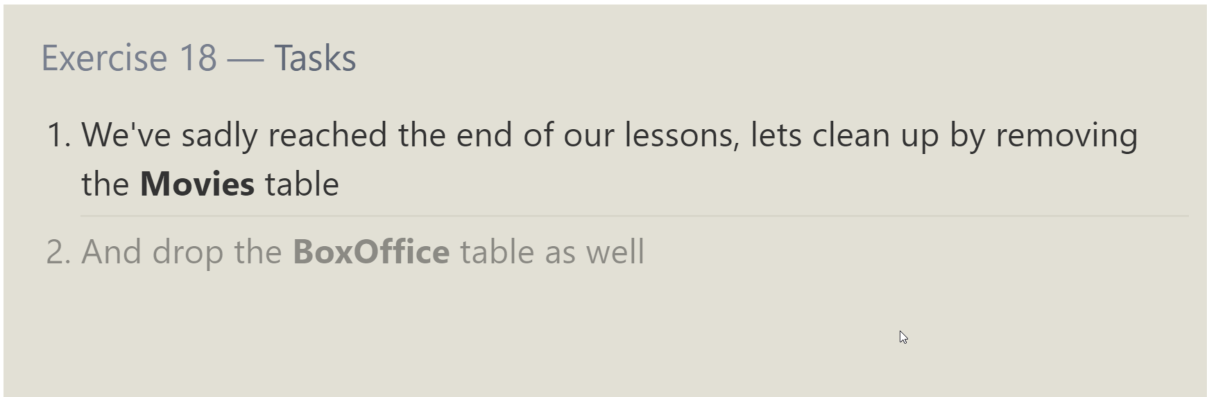
```
ALTER TABLE Movies
  ADD COLUMN Language TEXT DEFAULT "English";
```

Exercise 18

SQLBolt - Learn SQL - SQL Lesson 18: Dropping tables

In some rare cases, you may want to remove an entire table including all of its data and metadata, and to do so, you can use the DROP TABLE statement, which differs from the DELETE statement in that it also removes the table schema from the database entirely.

https://sqlbolt.com/lesson/dropping_tables


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

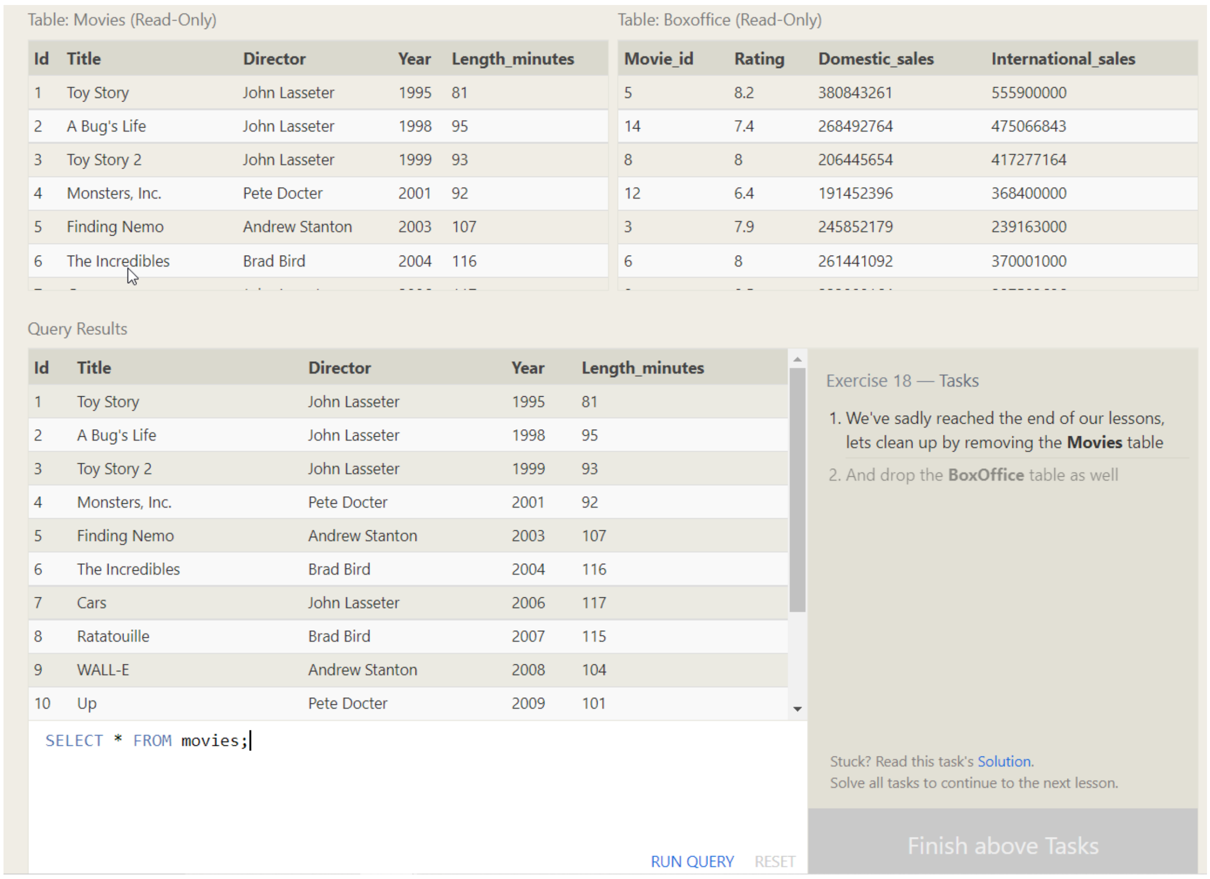


Table: Movies (Read-Only)

Id	Title	Director	Year	Length_minutes
1	Toy Story	John Lasseter	1995	81
2	A Bug's Life	John Lasseter	1998	95
3	Toy Story 2	John Lasseter	1999	93
4	Monsters, Inc.	Pete Docter	2001	92
5	Finding Nemo	Andrew Stanton	2003	107
6	The Incredibles	Brad Bird	2004	116

Table: Boxoffice (Read-Only)

Movie_id	Rating	Domestic_sales	International_sales
5	8.2	380843261	555900000
14	7.4	268492764	475066843
8	8	206445654	417277164
12	6.4	191452396	368400000
3	7.9	245852179	239163000
6	8	261441092	370001000

Query Results

Id	Title	Director	Year	Length_minutes
1	Toy Story	John Lasseter	1995	81
2	A Bug's Life	John Lasseter	1998	95
3	Toy Story 2	John Lasseter	1999	93
4	Monsters, Inc.	Pete Docter	2001	92
5	Finding Nemo	Andrew Stanton	2003	107
6	The Incredibles	Brad Bird	2004	116
7	Cars	John Lasseter	2006	117
8	Ratatouille	Brad Bird	2007	115
9	WALL-E	Andrew Stanton	2008	104
10	Up	Pete Docter	2009	101

```
SELECT * FROM movies;
```

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

Stuck? Read this task's Solution.
Solve all tasks to continue to the next lesson.

18(i). We've sadly reached the end of our lessons, lets clean up by removing the Movies table

```
DROP TABLE Movies;
```

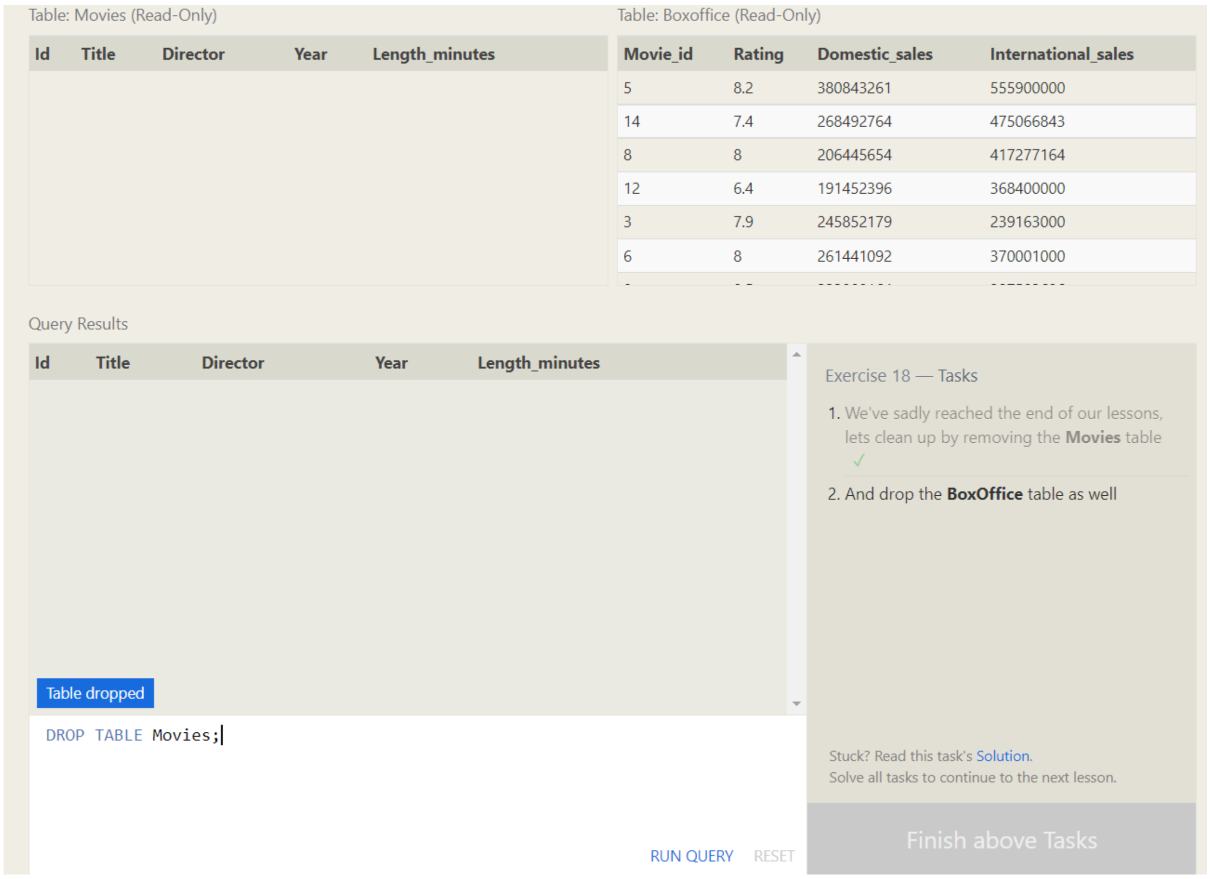


Table: Movies (Read-Only)

Id	Title	Director	Year	Length_minutes

Table: Boxoffice (Read-Only)

Movie_id	Rating	Domestic_sales	International_sales
5	8.2	380843261	555900000
14	7.4	268492764	475066843
8	8	206445654	417277164
12	6.4	191452396	368400000
3	7.9	245852179	239163000
6	8	261441092	370001000

Query Results

Id	Title	Director	Year	Length_minutes

Table dropped

```
DROP TABLE Movies;
```

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

Stuck? Read this task's Solution.
Solve all tasks to continue to the next lesson.

18(ii). And drop the BoxOffice table as well

```
DROP TABLE BoxOffice;
```

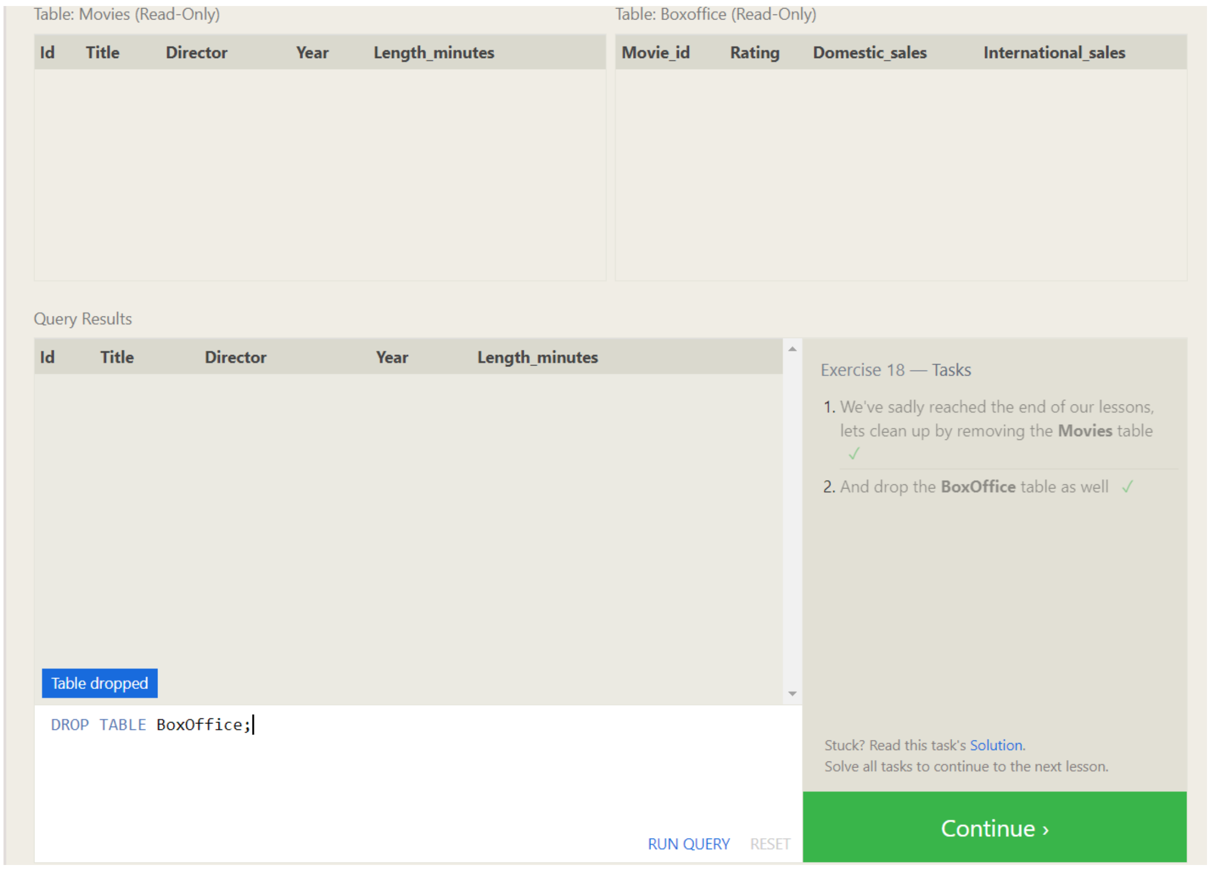


Table: Movies (Read-Only)

Id	Title	Director	Year	Length_minutes

Table: Boxoffice (Read-Only)

Movie_id	Rating	Domestic_sales	International_sales

Query Results

Id	Title	Director	Year	Length_minutes

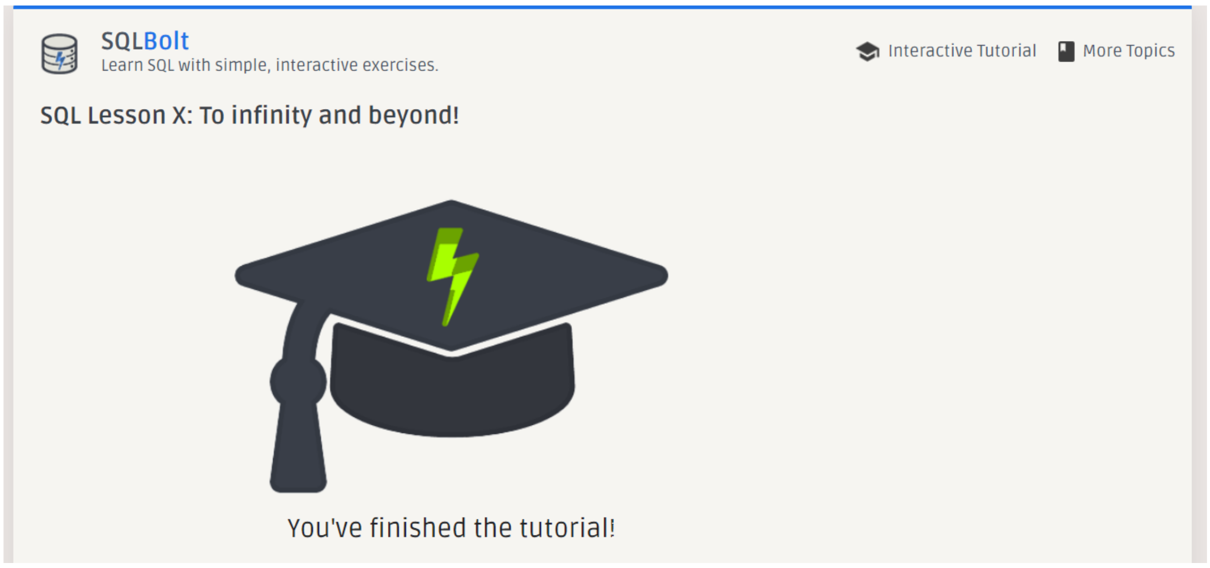
Table dropped

```
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 ✓

Stuck? Read this task's Solution.
Solve all tasks to continue to the next lesson.



SQL Lesson X: To infinity and beyond!

You've finished the tutorial!