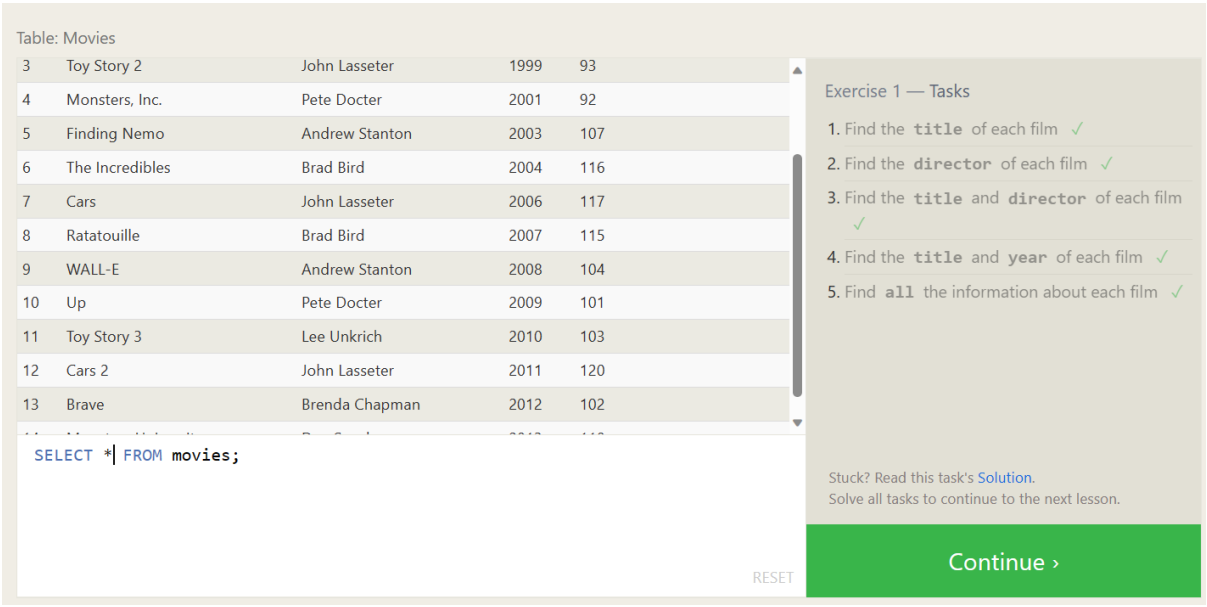
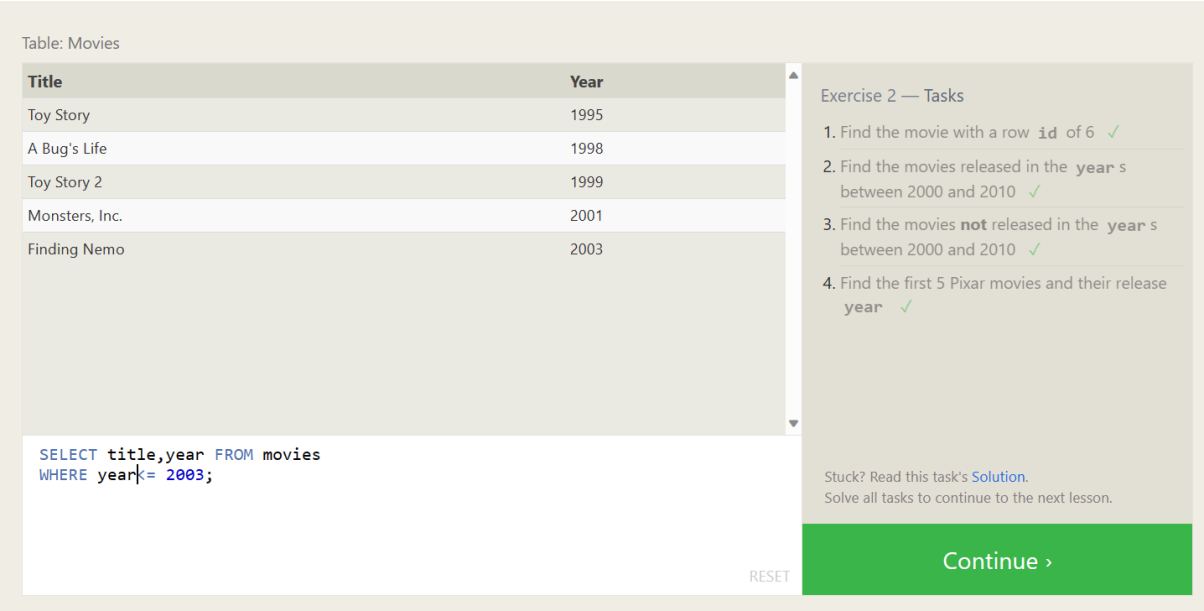
Exercise 1



Query

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;

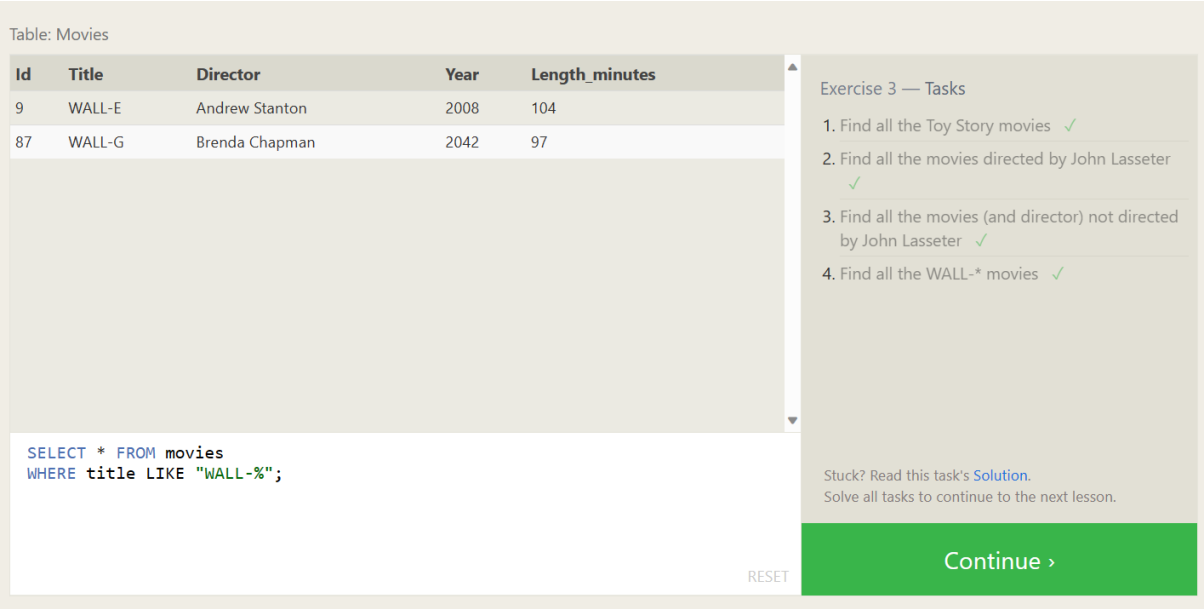
Exercise 2



Query

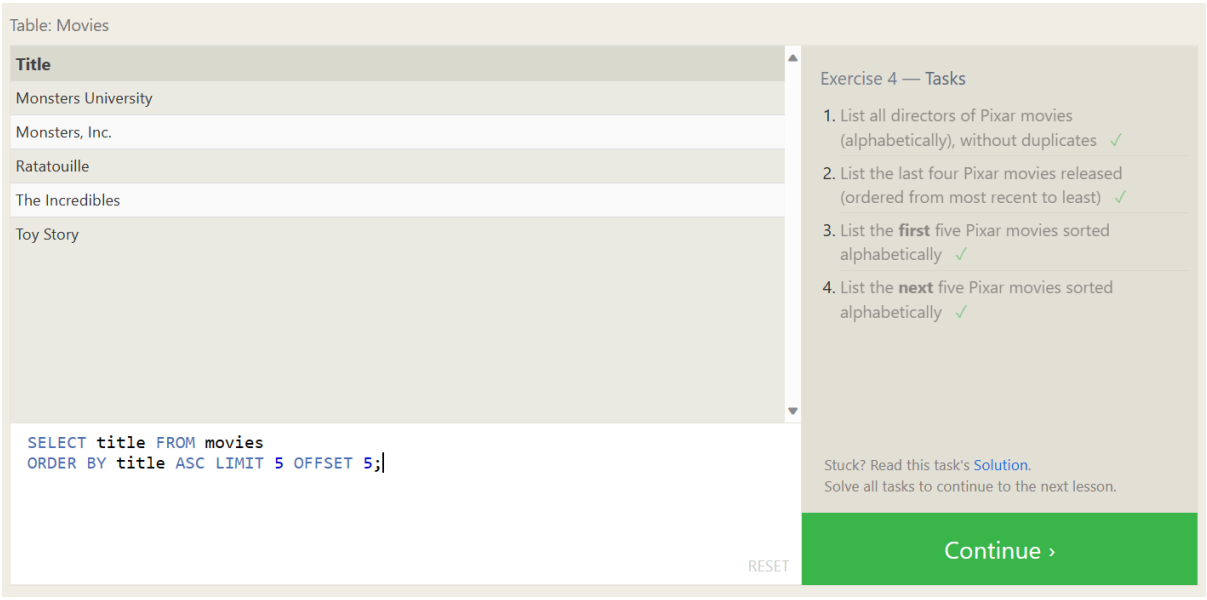
1. SELECT * FROM movies where id=6;
2. SELECT Title FROM movies where year BETWEEN 2000 AND 2010;
3. SELECT Title FROM movies where year NOT BETWEEN 2000 AND 2010;
4. SELECT * FROM movies where id IN(1,2,3,4,5)

Exercise 3



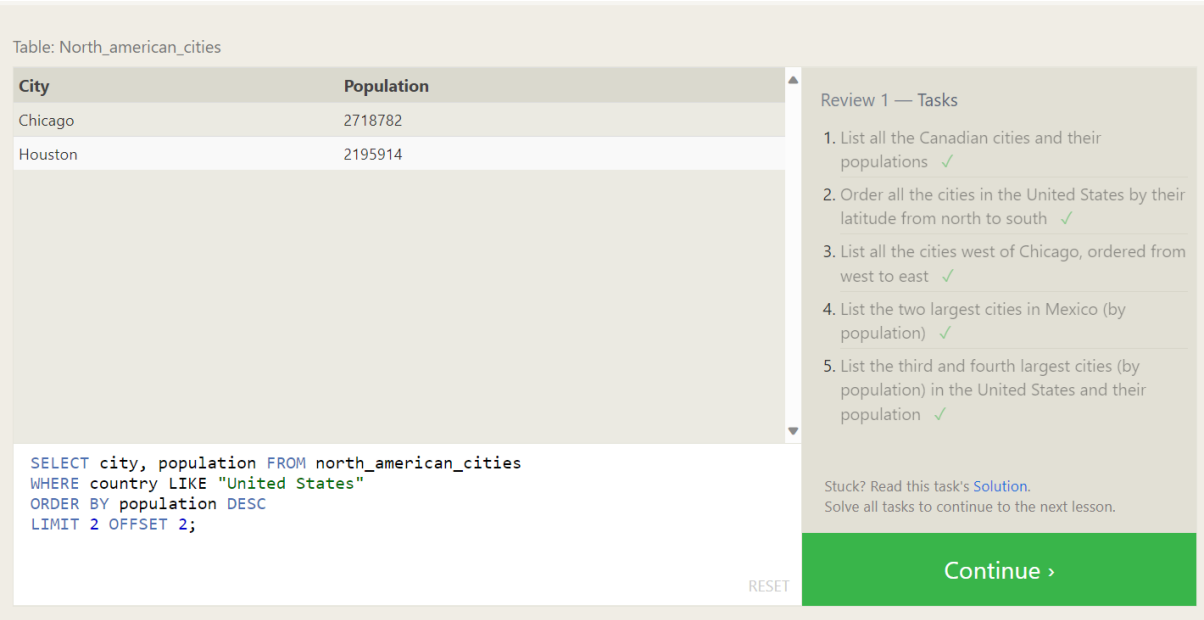
1. SELECT * FROM movies where title like "Toy%";
2. SELECT * FROM movies where director="John Lasseter";
3. SELECT title, director FROM movies where director!="John Lasseter";
4. SELECT * FROM movies where title like "WALL-%";

Exercise 4



1. SELECT DISTINCT director FROM movies order by director ASC;
2. SELECT year, title FROM movies order by year desc limit 4;
3. SELECT year, title FROM movies order by title ASC limit 5;
4. SELECT year, title FROM movies order by title ASC limit 5 offset 5;

Exercise 5



1. SELECT city,country, population FROM north_american_cities where country like "canada";

2. SELECT city, latitude FROM north_american_cities

   WHERE country = "United States"

   ORDER BY latitude DESC;

3. SELECT city, longitude FROM north_american_cities

   WHERE longitude < -87.629798

   ORDER BY longitude ASC;

4. SELECT city, population FROM north_american_cities

   WHERE country LIKE "Mexico"

   ORDER BY population DESC LIMIT 2;

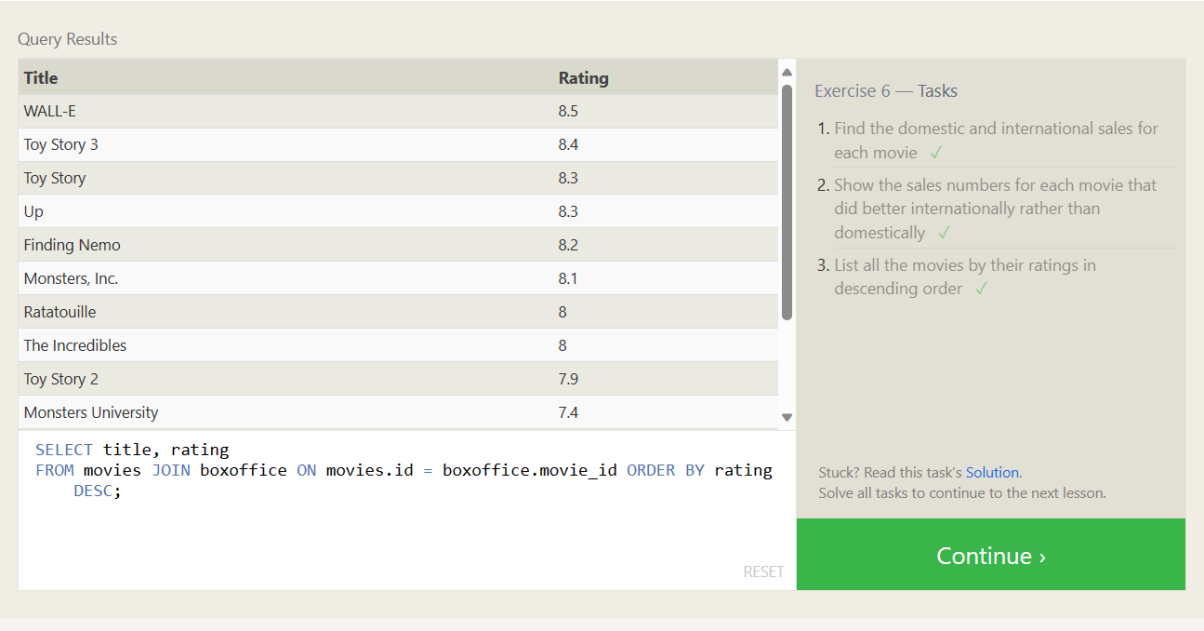
5. SELECT city, population FROM north_american_cities

   WHERE country LIKE "United States"

   ORDER BY population DESC

   LIMIT 2 OFFSET 2;

Exercise 6



1.SELECT title, domestic_sales, international_sales

FROM movies

JOIN boxoffice

ON movies.id = boxoffice.movie_id;

2.SELECT title, domestic_sales, international_sales

FROM movies

JOIN boxoffice

ON movies.id = boxoffice.movie_id

WHERE international_sales > domestic_sales;

3. SELECT title, rating

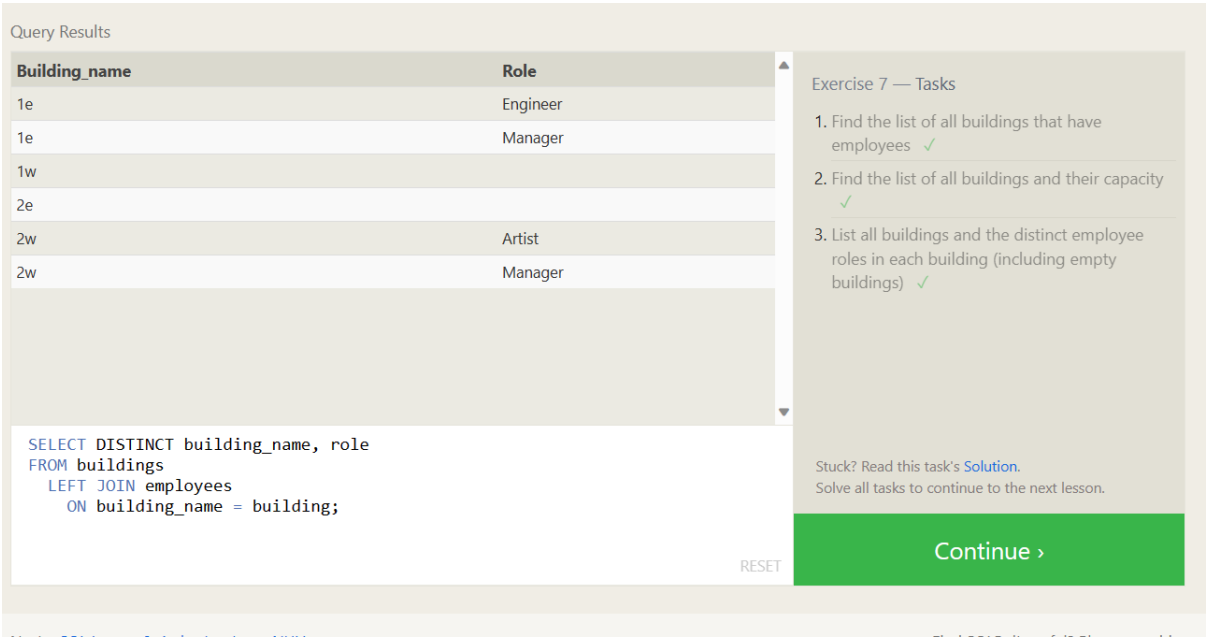
FROM movies

JOIN boxoffice

ON movies.id = boxoffice.movie_id

ORDER BY rating DESC

Exercise 7



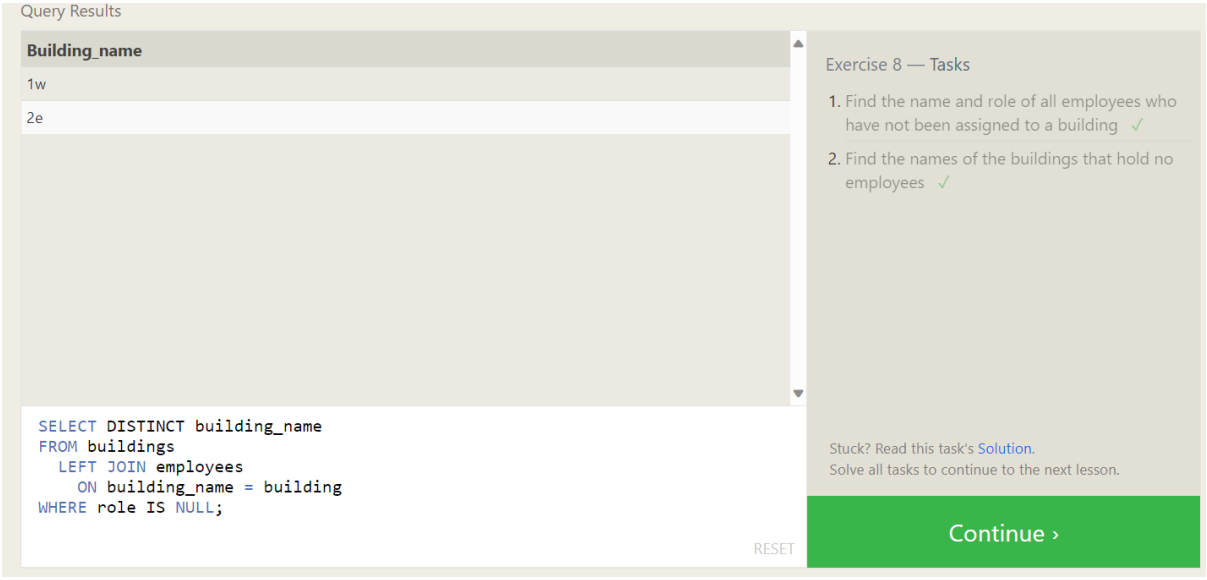
1. SELECT DISTINCT building FROM employees;
2. SELECT * FROM buildings;
3. SELECT DISTINCT building_name, role

   FROM buildings

   LEFT JOIN employees

   ON building_name = building;

Exercise 8



1. SELECT name, role FROM employees

   WHERE building IS NULL;

2. SELECT DISTINCT building_name

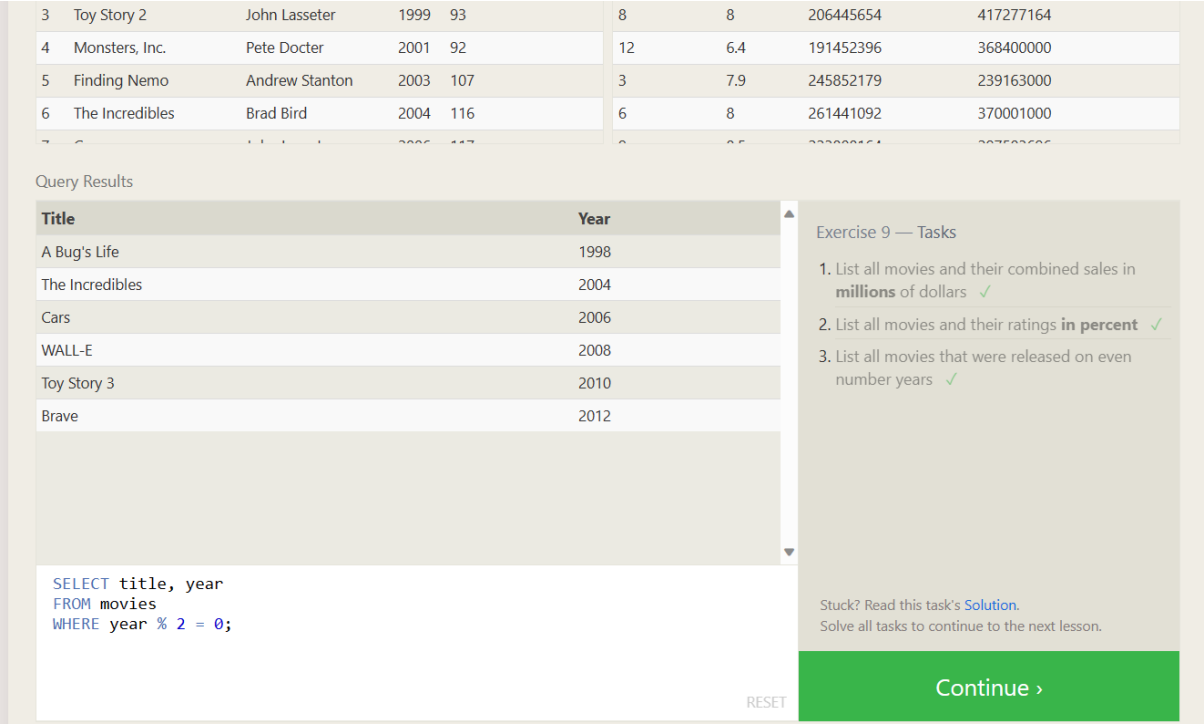
   FROM buildings

   LEFT JOIN employees

   ON building_name = building

   WHERE role IS NULL;

Exercise 9



3	Toy Story 2	John Lasseter	1999	93	8	8	206445654	417277164
4	Monsters, Inc.	Pete Docter	2001	92	12	6.4	191452396	368400000
5	Finding Nemo	Andrew Stanton	2003	107	3	7.9	245852179	239163000
6	The Incredibles	Brad Bird	2004	116	6	8	261441092	370001000

1. SELECT title, (domestic_sales + international_sales) / 1000000 AS gross_sales_millions

   FROM movies

   JOIN boxoffice

   ON movies.id = boxoffice.movie_id;

2. SELECT title, rating * 10 AS rating_percent

   FROM movies

   JOIN boxoffice

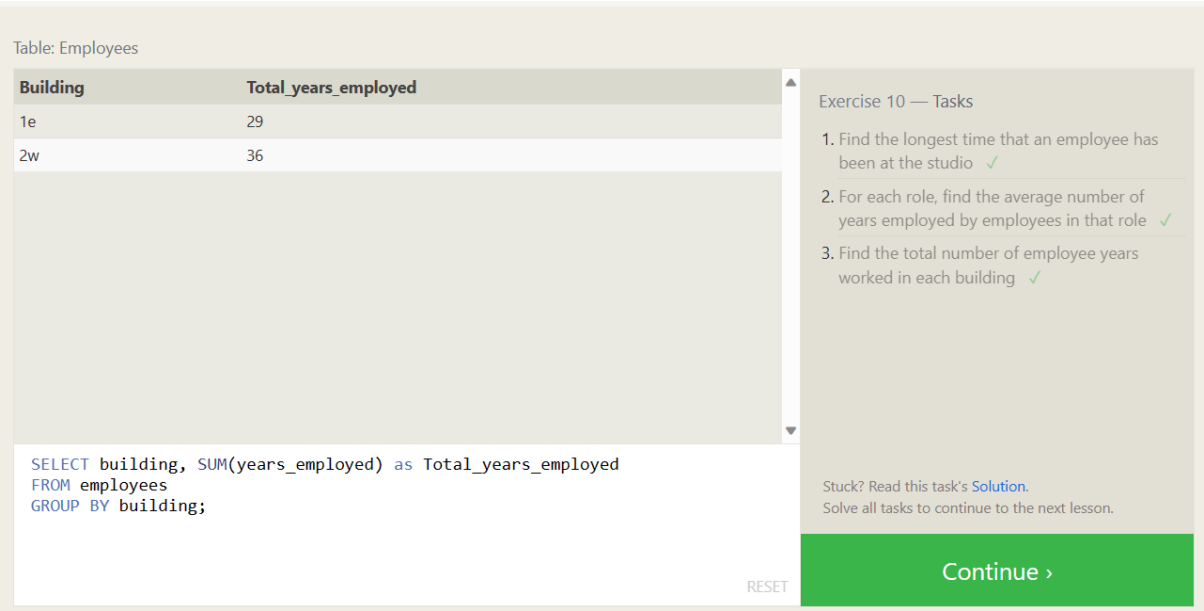
   ON movies.id = boxoffice.movie_id;

3. SELECT title, year

   FROM movies

   WHERE year % 2 = 0;

Exercise 10



1. SELECT MAX(years_employed) as Max_years_employed

   FROM employees;

2. SELECT role, AVG(years_employed) as Average_years_employed

   FROM employees

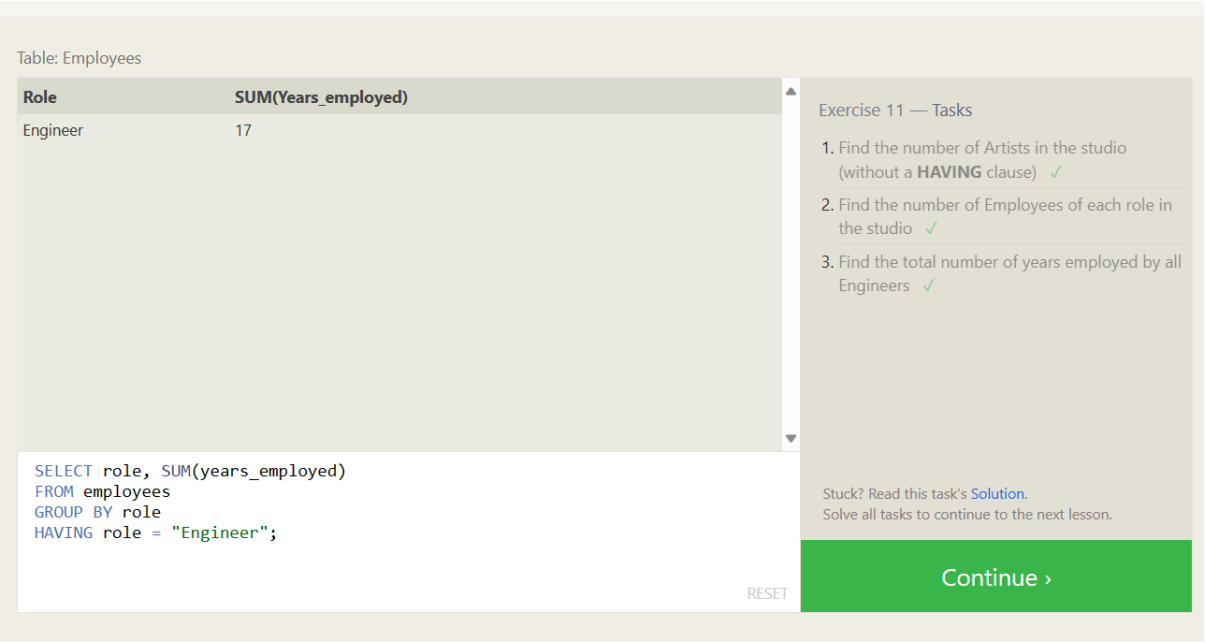
   GROUP BY role;

3. SELECT building, SUM(years_employed) as Total_years_employed

   FROM employees

   GROUP BY building;

Exercise 11



1. SELECT role, COUNT(*) as Number_of_artists

   FROM employees

   WHERE role = "Artist";

2. SELECT role, COUNT(*)

   FROM employees

   GROUP BY role;

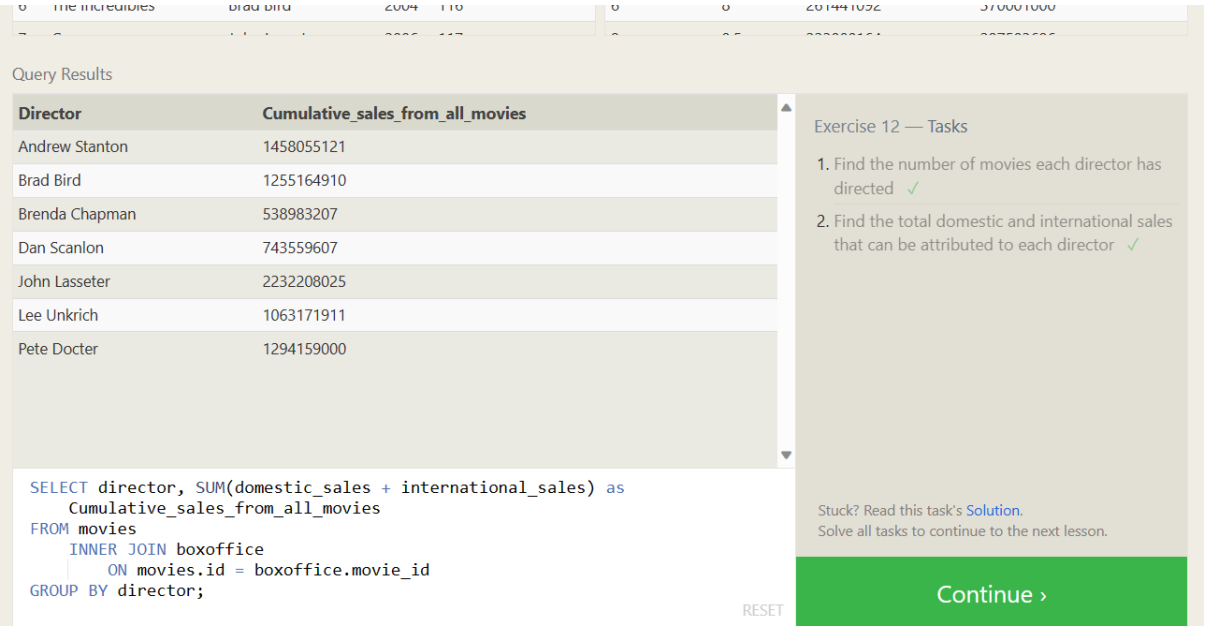
3. SELECT role, SUM(years_employed)

   FROM employees

   GROUP BY role

   HAVING role = "Engineer";

Exercise 12



Query Results

Director	Cumulative_sales_from_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
    Cumulative_sales_from_all_movies
FROM movies
    INNER JOIN boxoffice
        ON movies.id = boxoffice.movie_id
GROUP BY director;
```

RESET

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 ✓

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

Continue ›

1. SELECT director, COUNT(id) as Num_movies_directed

   FROM movies

   GROUP BY director;

2. SELECT director, SUM(domestic_sales + international_sales) as Cumulative_sales_from_all_movies

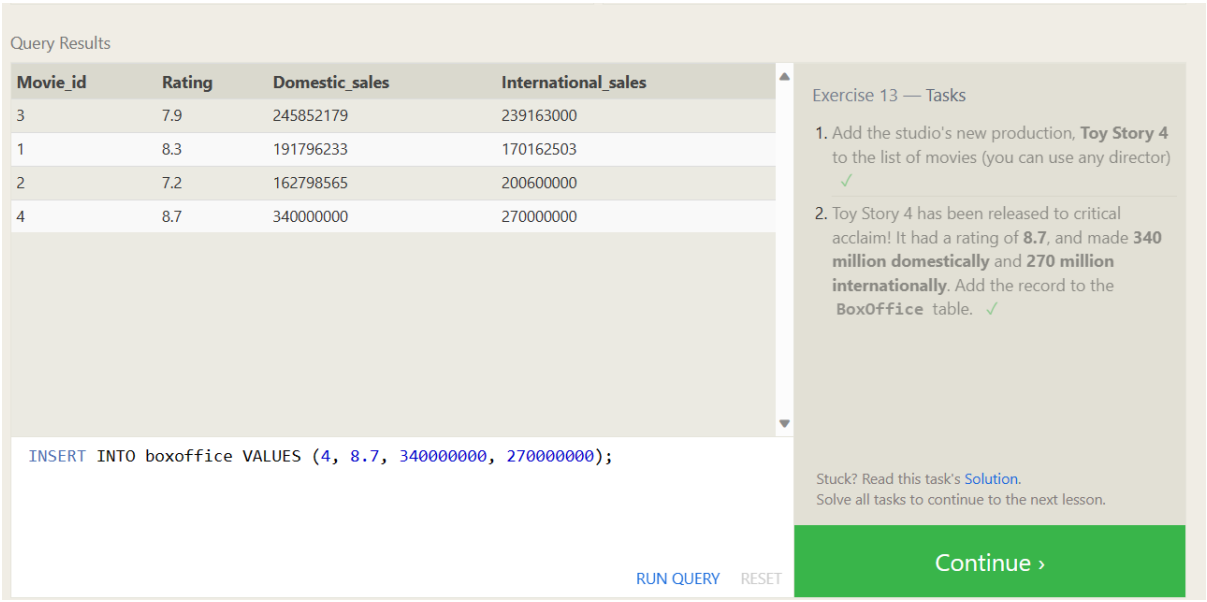
   FROM movies

   INNER JOIN boxoffice

   ON movies.id = boxoffice.movie_id

   GROUP BY director;

Exercise 13



1. INSERT INTO movies VALUES (4, "Toy Story 4", "El Directore", 2015, 90);
2. INSERT INTO boxoffice VALUES (4, 8.7, 340000000, 270000000);

Exercise 14

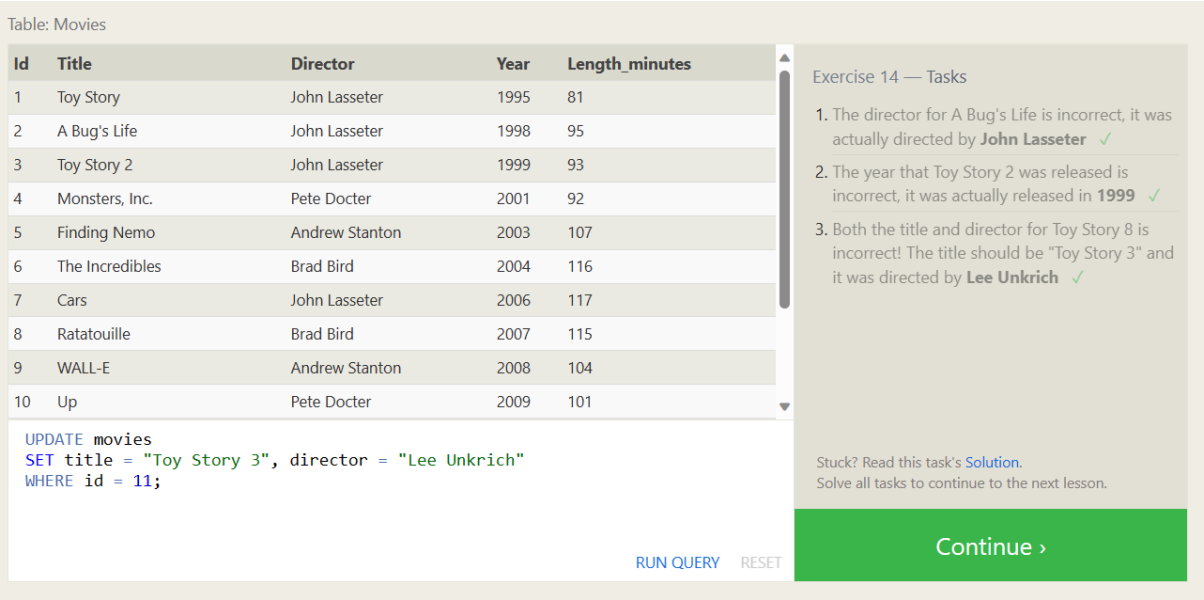


Table: Movies

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

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

RUN QUERY RESET

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** ✓

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

Continue ›

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;

Exercise 15

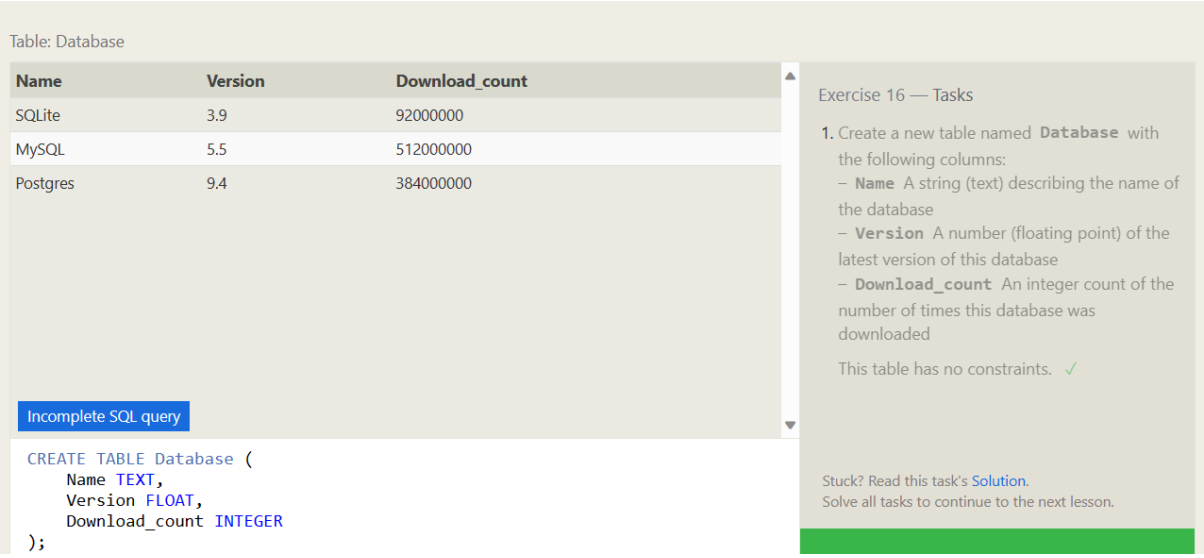
Table: Movies

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

1. DELETE FROM movies where year < 2005;

2. DELETE FROM movies where director = "Andrew Stanton";

Exercise 16



1. CREATE TABLE Database (

   Name TEXT,

   Version FLOAT,

   Download_count INTEGER);

Exercise 17

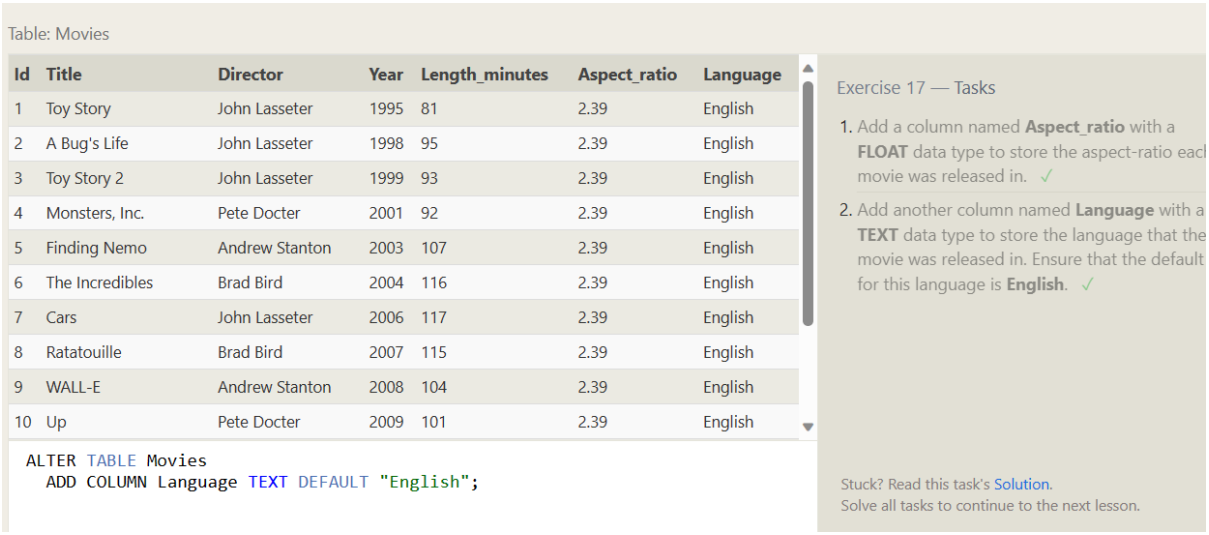


Table: Movies

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
10	Up	Pete Docter	2009	101	2.39	English

```
ALTER TABLE Movies
  ADD COLUMN Language TEXT 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**. ✓

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

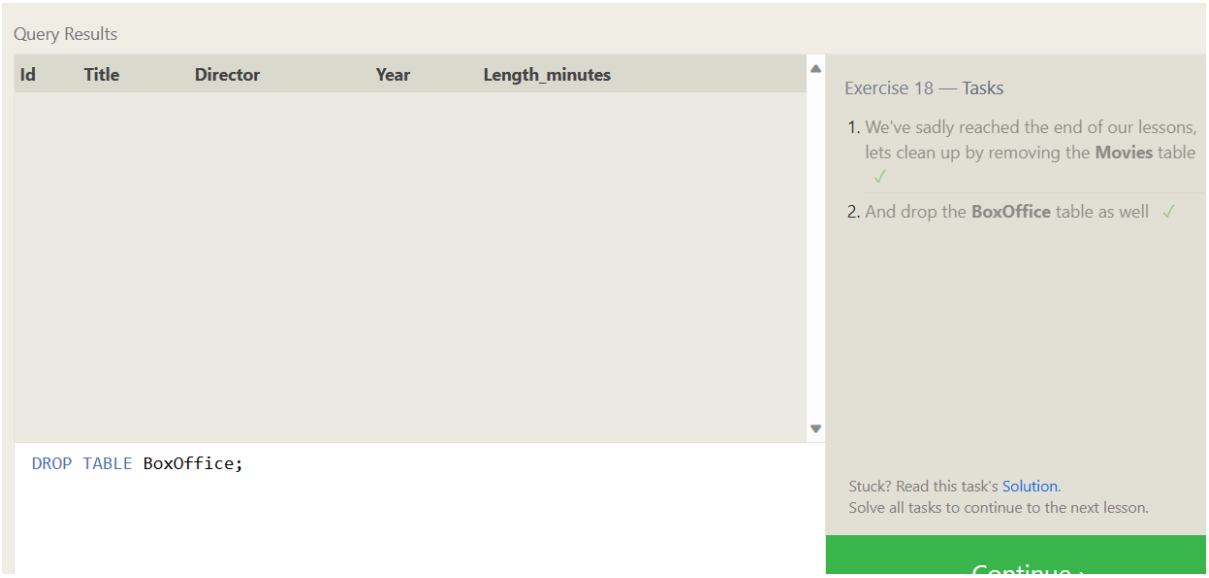
1. ALTER TABLE Movies

   ADD COLUMN Aspect_ratio FLOAT DEFAULT 2.39;

2. ALTER TABLE Movies

   ADD COLUMN Language TEXT DEFAULT "English";

Exercise 18



1. DROP TABLE Movies;
2. DROP TABLE BoxOffice;

SQLBolt
Learn SQL with simple, interactive exercises.

Interactive Tutorial | More Topics

SQL Lesson X: To infinity and beyond!

You've finished the tutorial!

We hope the lessons have given you a bit more experience with SQL and a bit more confidence to use SQL with your own data.

We've just brushed the surface of what SQL is capable of, so to get a better idea of how SQL can be used in the real world, we'll be adding more articles in the More Topics part of the site. If you have the time, we recommend that you continue to dive deeper into SQL!

If you need further details, it's also recommended that you read the documentation for the specific database that you are using, especially since each database has its own set of features and optimizations.

If you have any suggestions on how to make the site better, you can get in touch using one of the links in the footer below.

And if you found the lessons useful, please consider donating ($4) via Paypal to support our site. Your contribution will help keep the servers running and allow us to improve and add even more material in the future.