

# SQLBolt

## 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`

Table: Movies

<b>Id</b>	<b>Title</b>	<b>Director</b>	<b>Year</b>	<b>Length_minutes</b>
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 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 ✓

Stuck? Read this task's [Solution](#).  
Solve all tasks to continue to the next lesson.

RESET

Continue >

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

- 1 SELECT \* FROM movies where id = 6;
- 2 SELECT Title from movies WHERE year BETWEEN 2000 AND 2010;
  1. 3 SELECT Title from movies WHERE year NOT BETWEEN 2000 AND 2010;
- 4 SELECT \* FROM movies Limit 5;

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
	Incomplete SQL query	Pete Docter	2009	101

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` ✓

Stuck? Read this task's [Solution](#).  
Solve all tasks to continue to the next lesson.

RESET Continue ›

## 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-%';

Table: Movies

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

**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 ✓

```
SELECT * FROM movies where Title like 'WALL-%';
```

Stuck? Read this task's [Solution](#).  
Solve all tasks to continue to the next lesson.

RESET

**Continue ›**

## 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 ;

Table: Movies

<b>Id</b>	<b>Title</b>	<b>Director</b>	<b>Year</b>	<b>Length_minutes</b>
9	Monsters University	Dan Scanlon	2013	110
4	Monsters, Inc.	Pete Docter	2001	92
1	Ratatouille	Brad Bird	2007	115
14	The Incredibles	Brad Bird	2004	116
2	Toy Story	John Lasseter	1995	81

**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 ✓

```
SELECT *FROM movies ORDER BY Title ASC LIMIT 5 OFFSET 5 ;
```

Stuck? Read this task's [Solution](#).  
Solve all tasks to continue to the next lesson.

RESET

**Continue ›**

## SQL Review: Simple SELECT Queries

1. `SELECT city, population FROM north_american_cities WHERE country = '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 < ( SELECT longitude FROM north_american_cities WHERE city = 'Chicago' ) ORDER BY longitude ASC;`
4. `SELECT city, population FROM north_american_cities WHERE country = 'Mexico' ORDER BY population DESC LIMIT 2;`
5. `SELECT city, population FROM north_american_cities WHERE country = 'United States' ORDER BY population DESC LIMIT 2 OFFSET 2;`

Table: North\_american\_cities

City	Population
Chicago	2718782
Houston	2195914

### 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 ✓

Stuck? Read this task's [Solution](#).  
Solve all tasks to continue to the next lesson.

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

RESET

Continue >

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

1. `SELECT m.title, b.domestic_sales, b.international_sales FROM movies m JOIN boxoffice b ON m.id = b.movie_id;`
2. `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;`
3. `SELECT m.title, b.rating FROM movies m JOIN boxoffice b ON m.id = b.movie_id ORDER BY b.rating DESC;`

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
-	-	-	-	-

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

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 m.title, b.rating  
FROM movies m  
JOIN boxoffice b ON m.id = b.movie_id  
ORDER BY b.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 ✓

Stuck? Read this task's [Solution](#).  
Solve all tasks to continue to the next lesson.

RESET Continue >

## SQL Lesson 7: OUTER JOINS

1. SELECT DISTINCT building FROM employees;
2. SELECT building\_name, capacity FROM buildings;

Tables

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

Role	Name	Building	Years_employed
Engineer	Becky A.	1e	4
Engineer	Dan B.	1e	2
Engineer	Sharon F.	1e	6
Engineer	Dan M.	1e	4
Engineer	Malcom S.	1e	1
Artist	Tylar S.	2w	2

Query Results

Role	Name	Building	Capacity
Engineer	Sharon F.	1e	6
Engineer	Dan M.	1e	4
Engineer	Malcom S.	1e	1
Artist	Tylar S.	2w	2
Artist	Sherman D.	2w	8
Artist	Jakob J.	2w	6
Artist	Lillia A.	2w	7
Artist	Brandon J.	2w	7
Manager	Scott K.	1e	9
Manager	Shirlee M.	1e	3
Incomplete SQL query	Ia O.	2w	6

LEFT JOIN  
employees e ON b.building\_name = e.building  
GROUP BY  
b.building\_name  
ORDER BY  
b.building\_name;

RESET

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)

Stuck? Read this task's [Solution](#).  
Solve all tasks to continue to the next lesson.

Finish above Tasks

## SQL Lesson 8: A short note on NULLs

1. SELECT name, role FROM employees WHERE building IS NULL;
- 2.

This exercise will be a sort of review of the last few lessons. We're using the same `Employees` and `Buildings` table from the last lesson, but we've hired a few more people, who haven't yet been assigned a building.

Table: Buildings (Read-Only)		Table: Employees (Read-Only)			
Building_name	Capacity	Role	Name	Building	Years_employed
1e	24	Engineer	Becky A.	1e	4
1w	32	Engineer	Dan B.	1e	2
2e	16	Engineer	Sharon F.	1e	6
2w	20	Engineer	Dan M.	1e	4
		Engineer	Malcom S.	1e	1
		Artist	Tylar S.	2w	2
				-	-
				-	-

Query Results

Role	Name	Building	Years_employed
Engineer	Becky A.	1e	4
Engineer	Dan B.	1e	2
Engineer	Sharon F.	1e	6
Engineer	Dan M.	1e	4
Engineer	Malcom S.	1e	1
Artist	Tylar S.	2w	2
Artist	Sherman D.	2w	8
Artist	Jakob J.	2w	6
Artist	Lilia A.	2w	7
	No such column: building	In J.	7

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

Stuck? Read this task's [Solution](#).  
Solve all tasks to continue to the next lesson.

## SQL Lesson 9: Queries with expressions

1. 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;
2. SELECT m.title, b.rating \* 10 AS rating\_percent FROM movies m JOIN boxoffice b ON m.id = b.movie\_id;
3. SELECT title, year FROM movies WHERE year % 2 = 0;

Table: Movies (Read-Only)

<b>Id</b>	<b>Title</b>	<b>Director</b>	<b>Year</b>	<b>Length_minutes</b>
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)

<b>Movie_id</b>	<b>Rating</b>	<b>Domestic_sales</b>	<b>International_sales</b>
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

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

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 ✓

Stuck? Read this task's [Solution](#). Solve all tasks to continue to the next lesson.

[Next → SQL Lesson 10: Queries with aggregates \(Pt. 1\)](#)      [Find SQL Bolt useful? Please consider](#)

[RESET](#)      [Continue >](#)

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

1. `SELECT MAX(years_employed) AS longest_time_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_worked FROM employees GROUP BY building;`

```
FROM my_table  
WHERE constraint_expression  
GROUP BY column;
```

The `GROUP BY` clause works by grouping rows that have the same value in the column specified.

### Exercise

For this exercise, we are going to work with our `Employees` table. Notice how the rows in this table have shared data, which will give us an opportunity to use aggregate functions to summarize some high-level metrics about the teams. Go ahead and give it a shot.

Table: Employees

Building	Total_years_worked
1e	29
2w	36

#### 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 ✓

```
SELECT building,  
       SUM(years_employed) AS total_years_worked  
  FROM employees  
 GROUP BY building;  
;
```

Stuck? Read this task's [Solution](#).  
Solve all tasks to continue to the next lesson.

RESET

Continue >

Next - [SQL Lesson 11: Queries with aggregates \(Pt. 2\)](#)

Find SQLBolt useful? Please consider

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

1. SELECT COUNT(\*) AS number\_of\_artists FROM employees WHERE role = 'Artist';
2. SELECT role, COUNT(\*) AS number\_of\_employees FROM employees GROUP BY role;
3. SELECT SUM(years\_employed) AS total\_years\_employed FROM employees WHERE role = 'Engineer';

The `HAVING` clause constraints are written the same way as the `WHERE` clause constraints, and are applied to the grouped rows. With our examples, this might not seem like a particularly useful construct, but if you imagine data with millions of rows with different properties, being able to apply additional constraints is often necessary to quickly make sense of the data.

Did you know?

If you aren't using the '`GROUP BY`' clause, a simple '`WHERE`' clause will suffice.

### Exercise

For this exercise, you are going to dive deeper into `Employee` data at the film studio. Think about the different clauses you want to apply for each task.

Table: Employees

Total\_years\_employed

17

```
SELECT SUM(years_employed) AS total_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 ✓

Stuck? Read this task's [Solution](#).  
Solve all tasks to continue to the next lesson.

RESET

Continue >

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

1. SELECT m.director, COUNT(\*) AS number\_of\_movies FROM movies m GROUP BY m.director;

your understanding of queries, so don't be discouraged if you find them challenging. Just try your best.

Table: Movies (Read-Only)

<b>ID</b>	<b>Title</b>	<b>Director</b>	<b>Year</b>	<b>Length_minutes</b>
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)

<b>Movie_id</b>	<b>Rating</b>	<b>Domestic_sales</b>	<b>International_sales</b>
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

<b>Director</b>	<b>Total_domestic_sales</b>	<b>Total_international_sales</b>
Andrew Stanton	604651425	853403696
Brad Bird	467886746	787278164
Brenda Chapman	237283207	301700000
Dan Scanlon	268492764	475066843
John Lasseter	1035982355	1196225670
Lee Unkrich	415004880	648167031
Pete Docter	582920420	711238580

#### 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.

[Finish above Tasks](#)

```
SELECT m.director,
       SUM(b.domestic_sales) AS total_domestic_sales,
       SUM(b.international_sales) AS total_international_sales
  FROM movies m
 JOIN boxoffice b ON m.id = b.movie_id
 GROUP BY m.director;
```

RESET

## SQL Lesson 13: Inserting rows

1. `INSERT INTO movies (title, director, year, length_minutes) VALUES ('Toy Story 4', 'Josh Cooley', 2019, 100);`
2. `INSERT INTO boxoffice (movie_id, rating, domestic_sales, international_sales) VALUES (15, 8.7, 340000000, 270000000);`

Table: Movies (Read-Only)

<b>Id</b>	<b>Title</b>	<b>Director</b>	<b>Year</b>	<b>Length_minutes</b>
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	Josh Cooley	2019	100

Table: Boxoffice (Read-Only)

<b>Movie_id</b>	<b>Rating</b>	<b>Domestic_sales</b>	<b>International_sales</b>
3	7.9	245852179	239163000
1	8.3	191796233	170162503
2	7.2	162798565	200600000
11	8.7	340000000	270000000
11	8.7	340000000	270000000
15	8.7	340000000	270000000

Query Results

<b>Movie_id</b>	<b>Rating</b>	<b>Domestic_sales</b>	<b>International_sales</b>
3	7.9	245852179	239163000
1	8.3	191796233	170162503
2	7.2	162798565	200600000
11	8.7	340000000	270000000
11	8.7	340000000	270000000
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. ✓

Stuck? Read this task's [Solution](#).  
Solve all tasks to continue to the next lesson.

RUN QUERY RESET Continue >

## 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;

**TAKING CARE**

Most people working with SQL will make mistakes updating data at one point or another. Whether it's updating the wrong set of rows in a production database, or accidentally leaving out the `WHERE` clause (which causes the update to apply to *all* rows), you need to be extra careful when constructing `UPDATE` statements.

One helpful tip is to always write the constraint first and test it in a `SELECT` query to make sure you are updating the right rows, and only then writing the column/value pairs to update.

**Exercise**

It looks like some of the information in our `Movies` database might be incorrect, so go ahead and fix them through the exercises below.

Table: Movies

<b>Id</b>	<b>Title</b>	<b>Director</b>	<b>Year</b>	<b>Length_minutes</b>
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

|

**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.

[RUN QUERY](#) [RESET](#) [Continue >](#)

## SQL Lesson 15: Deleting rows

1. `DELETE FROM movies WHERE year < 2005;`
2. `DELETE FROM m`

**way to clear out a table completely (if intentional).**

**Taking extra care**

Like the `UPDATE` statement from last lesson, it's recommended that you run the constraint in a `SELECT` query first to ensure that you are removing the right rows. Without a proper backup or test database, it is downright easy to irrevocably remove data, so always read your `DELETE` statements twice and execute once.

**Exercise**

The database needs to be cleaned up a little bit, so try and delete a few rows in the tasks below.

Table: Movies

<b>Id</b>	<b>Title</b>	<b>Director</b>	<b>Year</b>	<b>Length_minutes</b>
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

|

**Exercise 15 – Tasks**

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

Stuck? Read this task's [Solution](#). Solve all tasks to continue to the next lesson.

[RUN QUERY](#) [RESET](#) [Continue >](#)

`ovies WHERE director = 'Andrew Stanton';`

## SQL Lesson 16: Creating tables

## 1. CREATE TABLE Database ( Name TEXT, Version FLOAT, Download\_count INT );

**Exercise**  
In this exercise, you'll need to create a new table for us to insert some new rows into.

Table: Database

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

**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. ✓

Stuck? Read this task's [Solution](#).  
Solve all tasks to continue to the next lesson.

**CREATE TABLE** Database ( Name `TEXT`, Version `FLOAT`, Download\_count `INT` );

[RUN QUERY](#) [RESET](#) [Continue ›](#)

Next - [SQL Lesson 17: Altering tables](#)  
Previous - [SQL Lesson 15: Deleting rows](#)

Find SQLBolt useful? Please consider [Donating \(\\$4\) via Paypal](#) to support our site.

## SQL Lesson 17: Altering tables

1. ALTER TABLE movies ADD COLUMN Aspect\_ratio FLOAT;
2. ALTER TABLE movies ADD COLUMN Language TEXT DEFAULT 'English';

**Other changes**

Each database implementation supports different methods of altering their tables, so it's always best to consult your database docs before proceeding: [MySQL](#), [Postgres](#), [SQLite](#), [Microsoft SQL Server](#).

**Exercise**

Our exercises use an implementation that only support adding new columns, so give that a try below.

Table: Movies

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

[RUN QUERY](#) [RESET](#) [Continue ›](#)

## SQL Lesson 18: Dropping tables

1. DROP TABLE movies;

2. DROP TABLE boxoffice;

Query Results

Id	Title	Director	Year	Length_minutes

|

RUN QUERY    RESET

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.

**Continue >**

Next - [SQL Lesson X: To infinity and beyond!](#)  
Previous - [SQL Lesson 17: Altering tables](#)

Find SQLBolt useful? Please consider [Donating \(\\$4\)](#) via [Paypal](#) to support our site.