

# SQL BOLT All Exercise

## Exercise-1

Problem 1: Find the title of each film

**SELECT title FROM Movies;**

Problem 2: Find the director of each film

**SELECT director FROM Movies;**

Problem 3: Find the title and director of each film

**SELECT title, director FROM Movies;**

Problem 4: Find the title and year of each film

**SELECT title, year FROM Movies;**

Problem 5: Find all the information about each film

**SELECT \* FROM Movies;**

The screenshot shows the SQL Bolt website interface. On the left, a table named 'Movies' is displayed with 5 columns: ID, Title, Director, Year, and Rating. The table contains 10 rows of data. On the right, a sidebar titled 'Exercise 1 — Tasks' lists five tasks, each with a green checkmark indicating completion. Below the tasks, there is a 'Continue' button. At the bottom of the page, there are navigation links for the next and previous lessons, and a donation prompt.

ID	Title	Director	Year	Rating
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
14	Monsters University	Dan Scanlon	2013	110

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.

[Continue >](#)

Next — SQL Lesson 2: Queries with constraints (Pt. 1)  
Previous — Introduction to SQL

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

## Exercise-2

Problem 1: Find the movie with a row id of 6

**SELECT title FROM Movies WHERE id = 6;**

Problem 2: Find the movies released in the years between 2000 and 2010

**SELECT title FROM Movies WHERE year BETWEEN 2000 AND 2010;**

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

**SELECT title FROM Movies WHERE year NOT BETWEEN 2000 AND 2010;**

Problem 4: Find the first 5 Pixar movies and their release year

**SELECT \* FROM movies WHERE id BETWEEN 1 AND 5;**

Using the right constraints, find the information we need from the **Movies** table for each task below.

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

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.

[Continue >](#)

Next — SQL Lesson 3: Queries with constraints (Pt. 2)  
Previous — SQL Lesson 1: SELECT queries 101

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

## Exercise-3

Problem 1: Find all the Toy Story movies

**SELECT title**

**FROM Movies**

**WHERE title LIKE 'Toy Story%';**

Problem 2: Find all the movies directed by John Lasseter

**SELECT title**

**FROM Movies**

**WHERE director = 'John Lasseter';**

Problem 3: Find all the movies (and directors) not directed by John Lasseter

**SELECT title, director**

**FROM Movies**

**WHERE director != 'John Lasseter';**

Problem 4: Find all the WALL-\* movies

**SELECT title**

**FROM Movies**

**WHERE title LIKE 'WALL-%';**

The screenshot shows the SQLBolt website interface. On the left, a table named 'Movies' is displayed with the following data:

Title
WALL-E
WALL-G

Below the table, a SQL query is entered in the editor:

```
SELECT title
FROM Movies
WHERE title LIKE 'WALL-%';
```

On the right, the 'Exercise 3 — Tasks' section lists four tasks, all of which are marked as completed with green checkmarks:

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 ✓

Below the tasks, there is a link to the 'Solution' and a 'Continue >' button. At the bottom of the page, there are navigation links for 'Next' and 'Previous' lessons, and a footer with social media links and a donation request.

## Exercise-4

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

**SELECT DISTINCT director FROM movies WHERE director IS NOT NULL ORDER BY director ASC;**

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

**SELECT title FROM movies WHERE director = 'Pixar' ORDER BY year DESC LIMIT 4;**

List the first five Pixar movies sorted alphabetically:

**SELECT title FROM movies WHERE director = 'Pixar' ORDER BY title ASC LIMIT 5;**

List the next five Pixar movies sorted alphabetically:

**SELECT title FROM movies WHERE director = 'Pixar' ORDER BY title ASC LIMIT 5 OFFSET 5;**

The screenshot shows the SQLBolt exercise interface. At the top, a message states: "we've gone and scrambled the **Movies** table for you in the exercise to better mimic what kind of data you might see in real life. Try and use the necessary keywords and clauses introduced above in your queries."

Below this, a table titled "Table: Movies" is displayed with the following data:

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

To the right of the table, a section titled "Exercise 4 — Tasks" lists four tasks, each with a green checkmark indicating completion:

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 ✓

Below the tasks, a text box contains the SQL query: `select title from movies order by title asc limit 5 offset 5;`. A "RESET" button is located to the right of the query box.

At the bottom right, a green "Continue >" button is visible.

At the bottom of the interface, there are links for "Next - SQL Review: Simple SELECT Queries" and "Previous - SQL Lesson 3: Queries with constraints (Pt. 2)". A footer message asks: "Find SQLBolt useful? Please consider Donating (\$4) via Paypal to support our site."

## Exercise-5

Problem 1: List all the Canadian cities and their populations

**SELECT city, Population FROM north\_american\_cities WHERE country = 'Canada';**

Problem 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;**

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

**SELECT city FROM North\_american\_cities WHERE Longitude < (SELECT Longitude FROM North\_american\_cities WHERE city = 'Chicago') ORDER BY Longitude ASC;**

Problem 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;**

Problem 5: 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 = 'United States' ORDER BY Population DESC LIMIT 2 OFFSET 2;**

The screenshot shows a web browser window with the URL `sqlbolt.com/lesson/select_queries_review`. The page content includes a table titled "Table: North\_american\_cities" with the following data:

City	Population
Chicago	2718782
Houston	2195914

Below the table is a SQL query editor with the following code:

```
SELECT City, Population
FROM North_american_cities
WHERE Country = 'United States'
ORDER BY Population DESC
LIMIT 2 OFFSET 2;
```

To the right of the query editor is a "Review 1 — Tasks" section with five items, each marked with a green checkmark:

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 ✓

Below the tasks is a green button labeled "Continue >". At the bottom of the page, there are links for "Next - SQL Lesson 6: Multi-table queries with JOINS" and "Previous - SQL Lesson 4: Filtering and sorting Query results", along with a "Find SQLBolt useful? Please consider Donating (\$4) via Paypal to support our site." link.

## Exercise-6

Problem 1: Find the domestic and international sales for each movie

**SELECT title, domestic\_sales, international\_sales FROM movies INNER JOIN boxoffice ON movies.id = boxoffice.movie\_id;**

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

**SELECT \* FROM movies INNER JOIN boxoffice ON movies.id = boxoffice.movie\_id WHERE international\_sales > domestic\_sales;**

Problem 3: List all the movies by their ratings in descending order

**SELECT title, rating FROM movies INNER JOIN boxoffice ON movies.id = boxoffice.movie\_id ORDER BY rating DESC;**

The screenshot shows the SQLBolt web application interface. At the top, there's a table with movie data. Below it, the 'Query Results' section displays a list of movies sorted by rating in descending order. The 'Exercise 6 — Tasks' section on the right lists three tasks, all of which are marked as completed. At the bottom, there's a 'Continue' button and a footer with navigation links and a donation request.

id	title	director	year	num_critic_reviews	rating	domestic_sales	international_sales
5	Finding Nemo	Andrew Stanton	2003	107	3	7.9	245852179
6	The Incredibles	Brad Bird	2004	116	6	8	261441092

Query Results

title	rating
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
Cars	7.2
A Bug's Life	7.2
Brave	7.2
Cars 2	6.4

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.

[Continue >](#)

Next — [SQL Lesson 7: OUTER JOINS](#)  
Previous — [SQL Review: Simple SELECT Queries](#)

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

## Exercise-7

Problem 1: Find the list of all buildings that have employees

**SELECT DISTINCT building\_name FROM buildings LEFT JOIN employees ON buildings.building\_name = employees.building WHERE building IS NOT NULL;**

Problem 2: Find the list of all buildings and their capacity

**SELECT \* FROM buildings;**

Problem 3: 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 buildings.building\_name = employees.building;**

Artist Tylar S. 2w 2

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 buildings.building_name = employees.building;
```

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.

Continue >

Next – SQL Lesson 8: A short note on NULLs  
Previous – SQL Lesson 6: Multi-table queries with JOINS

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

## Exercise-8

Find the Name and Role of Employees Not Assigned to a Building:

**SELECT Name, Role FROM Employees WHERE Building IS NULL;**

Find the Names of Buildings That Hold No Employees:

**SELECT Building\_name FROM Buildings LEFT JOIN Employees ON Buildings.Building\_name = Employees.Building WHERE Name IS NULL;**

The screenshot shows the SQLBolt website interface. At the top, there's a table with employee data:

Employee	Name	Role	Building
Engineer	Dan M.	1e	4
Engineer	Malcom S.	1e	1
Artist	Tylar S.	2w	2

Below this, the 'Query Results' section shows the output of the query: 'Building\_name' with values '1w' and '2e'.

The 'Exercise 8 — Tasks' section contains two 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. ✓

Below the tasks, there's a 'Continue' button and a 'Solution' link.

The SQL query entered in the editor is:

```
SELECT Building_name
FROM Buildings
LEFT JOIN Employees
ON Buildings.Building_name = Employees.Building
WHERE Name IS NULL;
```

At the bottom, there are links for 'Next - SQL Lesson 9: Queries with expressions' and 'Previous - SQL Lesson 7: OUTER JOINS'. A footer message asks to 'Find SQLBolt useful? Please consider Donating (\$4) via Paypal to support our site.'



## Exercise-9

List all movies and their combined sales in millions of dollars:

```
SELECT Title, (Domestic_sales + International_sales) / 1000000 AS "Combined Sales"
FROM Movies INNER JOIN Boxoffice ON Movies.Id = Boxoffice.Movie_id;
```

List all movies and their ratings in percent:

```
SELECT Title, ROUND((Rating / 10), 2) * 100 AS Ratings FROM Movies INNER JOIN
Boxoffice ON Movies.Id = Boxoffice.Movie_id ORDER BY Ratings DESC;
```

List all movies that were released on even-numbered years:

```
SELECT Title FROM Movies INNER JOIN Boxoffice ON Movies.Id =
Boxoffice.Movie_id WHERE Year % 2 = 0;
```

The screenshot shows the SQLBolt web application interface. At the top, there's a table with movie data. Below it, the 'Query Results' section displays a list of movie titles: The Incredibles, WALL-E, Toy Story 3, Cars, A Bug's Life, and Brave. To the right, the 'Exercise 9 — Tasks' section lists three tasks, all of which are marked as completed with green checkmarks:

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 ✓

Below the tasks, there's a 'Continue' button and a 'RESET' button. At the bottom, there are links for 'Next - SQL Lesson 10: Queries with aggregates (Pt. 1)' and 'Previous - SQL Lesson 8: A short note on NULLs'. A footer message encourages users to find SQLBolt useful and consider donating via PayPal.

## Exercise-10

Find the longest time that an employee has been at the studio:

```
SELECT Name, SUM(Years_employed) AS "Longest Time" FROM Employees  
GROUP BY Name ORDER BY "Longest Time" DESC LIMIT 1;
```

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;
```

Find the total number of employee years worked in each building:

```
SELECT Building, SUM(Years_employed) AS "Total Employee Years" FROM  
Employees GROUP BY Building;
```

The screenshot shows the SQLBolt website interface for Exercise 10. It includes a table of employee years by building, a list of three tasks, a SQL query editor with the query for total employee years by building, and navigation links.

Table: Employees

Building	Total Employee Years
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 ✓

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

[Continue >](#)

Next — [SQL Lesson 11: Queries with aggregates \(Pt. 2\)](#)  
Previous — [SQL Lesson 9: Queries with expressions](#)

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

[https://sqlbolt.com/lesson/select\\_queries\\_with\\_aggregates\\_pt2](https://sqlbolt.com/lesson/select_queries_with_aggregates_pt2)

## Exercise-11

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

```
SELECT COUNT(Role) AS "Number of Artists" FROM Employees WHERE Role = 'Artist';
```

Find the number of Employees of each role in the studio:

```
SELECT Role, COUNT(Role) AS "Number of Employees" FROM Employees GROUP BY Role;
```

Find the total number of years employed by all Engineers:

```
SELECT SUM(Years_employed) AS "Total Years Employed by Engineers" FROM Employees WHERE Role = 'Engineer' GROUP BY Role;
```

The screenshot shows a web browser window with the URL `sqlbolt.com/lesson/select_queries_with_aggregates_pt2`. A notification at the top states: "If you aren't using the 'GROUP BY' clause, a simple 'WHERE' clause will suffice." The main content area is titled "Exercise" and contains the instruction: "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." Below this, a table titled "Table: Employees" is shown with a single row: "Total Years Employed By Engineers" with the value "17". To the right, a panel titled "Exercise 11 — Tasks" lists three tasks, all of which are marked as completed with green checkmarks: 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 ✓, and 3. Find the total number of years employed by all Engineers ✓. At the bottom of the panel, there is a link to "Stuck? Read this task's Solution." and a note to "Solve all tasks to continue to the next lesson." A green "Continue" button is located at the bottom right of the panel. The SQL query entered in the editor is: 

```
SELECT SUM(Years_employed) AS "Total Years Employed by Engineers" FROM Employees WHERE Role = 'Engineer' GROUP BY Role;
```

## Exercise-12

Find the number of movies each director has directed:

**SELECT Director, COUNT(Title) AS "Number of Movies" FROM Movies GROUP BY Director;**

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

**SELECT Director, SUM(Domestic\_sales + International\_sales) AS "Total Sales" FROM Movies INNER JOIN Boxoffice ON Movies.Id = Boxoffice.Movie\_id GROUP BY Director;**

5 Finding Nemo Andrew Stanton 2003 107 3 7.9 245852179 239163000

6 The Incredibles Brad Bird 2004 116 6 8 261441092 370001000

Query Results

Director	Total Sales
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"
FROM Movies
INNER JOIN Boxoffice ON Movies.Id = Boxoffice.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 ✓

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

[Continue >](#)

Next – SQL Lesson 13: Inserting rows  
Previous – SQL Lesson 11: Queries with aggregates (Pt. 2)

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

## Exercise-13

Add the studio's new production, Toy Story 4 to the list of movies:

**INSERT INTO Movies (Title, Director) VALUES ('Toy Story 4', 'Vishnu');**

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 (Movie\_id, Rating, Domestic\_sales, International\_sales) VALUES (15, 8.7, 340000000, 270000000);**

The screenshot shows the SQLBolt website interface. On the left, a table titled 'Query Results' displays data for the 'Boxoffice' table. The table has four columns: 'Movie\_id', 'Rating', 'Domestic\_sales', and 'International\_sales'. It contains four rows of data, with the last row (Movie\_id 15) matching the values specified in the exercise instructions. On the right, a sidebar titled 'Exercise 13 — Tasks' lists two tasks. Task 1 is 'Add the studio's new production, Toy Story 4 to the list of movies (you can use any director)' and Task 2 is '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.' Both tasks are marked as completed with green checkmarks. Below the tasks, there is a 'Continue' button. At the bottom of the page, there are links for 'Next' and 'Previous' lessons, a 'Like' button, and a 'Post' button. The Windows taskbar is visible at the bottom of the screen.

Movie_id	Rating	Domestic_sales	International_sales
3	7.9	245852179	239163000
1	8.3	191796233	170162503
2	7.2	162798565	200600000
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.

[Continue >](#)

Next — SQL Lesson 14: Updating rows  
Previous — SQL Lesson 12: Order of execution of a Query

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

## Exercise-14

Update the director for "A Bug's Life" to be John Lasseter:

**UPDATE Movies SET Director = 'John Lasseter' WHERE Title = 'A Bug's Life';**

Update the release year for "Toy Story 2" to be 1999:

**UPDATE Movies SET Year = 1999 WHERE Title = 'Toy Story 2';**

Update the title and director for "Toy Story 8" to be "Toy Story 3" directed by Lee Unkrich:

**UPDATE Movies SET Title = 'Toy Story 3', Director = 'Lee Unkrich' WHERE Title = 'Toy Story 8';**

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

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 Title = 'Toy Story 8';

[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 >](#)

Next – [SQL Lesson 15: Deleting rows](#)  
Previous – [SQL Lesson 13: Inserting rows](#)

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

## Exercise-15

Remove all movies that were released before 2005:

**DELETE FROM Movies WHERE Year < 2005;**

Remove all movies directed by Andrew Stanton:

**DELETE FROM Movies WHERE Director = 'Andrew Stanton';**

The screenshot shows the SQLBolt website interface for Exercise 15. The main content area displays a table named 'Movies' with the following data:

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

Below the table is a text input field with a cursor. To the right of the table, there are two tasks for Exercise 15:

- 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. ✓

At the bottom of the task list, there is a link to the solution: "Stuck? Read this task's [Solution](#). Solve all tasks to continue to the next lesson." Below this is a green "Continue" button.

At the bottom of the page, there are links for "Next - SQL Lesson 16: Creating tables" and "Previous - SQL Lesson 14: Updating rows". There is also a footer with social media links and a donation request: "Find SQLBolt useful? Please consider Donating (\$4) via [Paypal](#) to support our site."

## Exercise-16

- **Problem 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(20), Version FLOAT, Download\_count int);**

The screenshot shows the SQLBolt interface for Exercise 16. At the top, there's a navigation bar with the URL 'sqlbolt.com/lesson/creating\_tables'. Below it, the exercise title 'Exercise 16' is displayed. The main content area contains a table named 'Database' with the following data:

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

Below the table, the SQL command to create the table is shown: `Create table Database(Name Varchar(20), Version FLOAT, Download_count int);`. To the right of the table, there's a sidebar with the exercise instructions: '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. ✓'. At the bottom of the sidebar, there's a 'Continue' button. The bottom of the interface shows a footer with 'Next - SQL Lesson 17: Altering tables' and 'Find SQLBolt useful? Please consider'.



## Exercise-17

Problem 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;`**

Problem 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(20) DEFAULT 'English';`**

Exercise

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

Table: Movies

Id	Title	Director	Year	Length_minutes	Aspect_ratio	Language
1	Toy Story	John Lasseter	1995	81		'English'
2	A Bug's Life	John Lasseter	1998	95		'English'
3	Toy Story 2	John Lasseter	1999	93		'English'
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'

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](#)

Next - [SQL Lesson 18: Dropping tables](#)  
Previous - [SQL Lesson 16: Creating tables](#)

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

## Exercise-18

Drop the Movies table:

**DROP TABLE Movies;**

Drop the BoxOffice table:

**DROP TABLE BoxOffice;**

The screenshot shows the SQLBolt web application interface. The browser address bar displays 'sqlbolt.com/lesson/dropping\_tables'. The main content area is divided into two columns. The left column contains a 'Query Results' section with a table header: 

Id	Title	Director	Year	Length_minutes
----	-------	----------	------	----------------

. Below the header, the table is empty. At the bottom of the left column, there is a text input field containing the SQL command 'DROP TABLE BoxOffice;' and two buttons: 'RUN QUERY' and 'RESET'. The right column contains a section titled 'Exercise 18 — Tasks' with a list of two tasks: '1. We've sadly reached the end of our lessons, lets clean up by removing the **Movies** table' (marked with a green checkmark) and '2. And drop the **BoxOffice** table as well' (also marked with a green checkmark). Below the tasks, there is a link for 'Solution' and a note: 'Stuck? Read this task's Solution. Solve all tasks to continue to the next lesson.' At the bottom of the right column is a green 'Continue >' button. The footer of the page includes the text 'Next - SQL Lesson X: To infinity and beyond!' and 'Find SQLBolt useful? Please consider'.