

# Lesson 1

first exercise will only involve the **Movies** table, and the default query below currently shows all the properties of each movie. To continue onto the next lesson, alter the query to find the exact information we need for each task.

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
11	Toy Story 3	Lee Unkrich	2010	103

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.

`SELECT * FROM movies;`

Continue >

Next — [SQL Lesson 2: Queries with constraints \(Pt. 1\)](#)

Find SQL Bolt useful? Please consider

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;

# Lesson 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	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` ✓

SELECT \* FROM movies where id between 1 and 5

RESET

Continue >

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

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

1.SELECT \* FROM movies where id=6;

2.SELECT \* FROM movies where year between 2000 and 2010

3.SELECT \* FROM movies where year not between 2000 and 2010

4.SELECT \* FROM movies where id between 1 and 5

## Lesson 3

SQLBolt - Learn SQL - SQL Lesson 3: Filtering and sorting Query results

Table: Movies

Id	Title	Director	Year	Length_minutes
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%'

Continue >

Next - SQL Lesson 4: Filtering and sorting Query results  
Previous - SQL Lesson 2: Queries with constraints (Pt. 1)

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

1.SELECT \* FROM movies where title like '%toy%'

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%'

## Lesson 4

There are a few concepts in this lesson, but all are pretty straight-forward to apply. To spice things up, 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.

Table: Movies

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

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

RESET

Continue >

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

- 1.SELECT distinct director FROM movies ORDER BY director ASC;
- 2.SELECT title, year FROM movies ORDER BY year DESC LIMIT 4;
- 3.SELECT title FROM movies ORDER BY title ASC LIMIT 5;
- 4.SELECT title FROM movies ORDER BY title ASC LIMIT 5 OFFSET 5;

## Lesson 5

different combination of clauses in your query for each task. Once you're done, continue onto the next lesson to learn about queries that span multiple tables.

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;
```

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.

Continue >

Next - SQL Lesson 6: Multi-table queries with JOINS

Find SQLBolt useful? Please consider

- 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, latitude FROM north\_american\_cities WHERE country = "United States"  
ORDER BY latitude DESC;
- 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;

## Lessons 6

The screenshot shows the SQLBolt website interface. At the top, there's a navigation bar with the site name and a search bar. Below it, a list of lessons is visible. The main content area is titled 'Query Results' and shows a table with movie titles and their ratings. The table is as follows:

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

Below the table, there's a code editor with the following SQL query:

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

To the right of the code editor, there's a section titled 'Exercise 6 — Tasks' with three 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 ✓

At the bottom of the exercise section, there's a green button labeled 'Continue >'. Below the exercise section, there's a footer with links to the next and previous lessons, and a donation link.

1. `SELECT title, domestic_sales, international_sales FROM movies JOIN boxoffice ON movies.id = boxoffice.movie_id;`
2. `SELECT title, domestic_sales, international_sale FROM movies JOIN boxoffice ON movies.id = boxoffice.movie_id WHERE international_sales > domestic_sale;`
3. `SELECT title, rating FROM movies JOIN boxoffice ON movies.id = boxoffice.movie_id ORDER BY rating DESC;`

## Lesson 7

The screenshot shows a web browser window with the URL `sqlbolt.com/lesson/select_queries_with_outer_joins`. The browser's address bar and tabs are visible at the top. The main content area is divided into several sections:

- Table:** A table with 4 columns. The first two columns are 'Building\_name' and 'Role'. The last two columns are 'Capacity' and 'Count'. The data is as follows:

Building_name	Role	Capacity	Count
1e	Engineer	1e	4
1e	Manager	1e	1
2w	Artist	2w	2
2w	Manager	2w	2
- Query Results:** A table with 2 columns: 'Building\_name' and 'Role'. The data is as follows:

Building_name	Role
1e	Engineer
1e	Manager
1w	
2e	
2w	Artist
2w	Manager
- SQL Query Input:** A text area containing the SQL query: `SELECT DISTINCT building_name, role FROM buildings LEFT JOIN employees ON building_name = building;`
- Exercise 7 — Tasks:** A list of three 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) ✓
- Buttons:** A 'RESET' button and a green 'Continue >' button.

The bottom of the browser window shows the Windows taskbar with the date '08-04-2023' and time '23:26'.

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;

## Lesson 8

The screenshot shows the SQLBolt interface for Lesson 8. At the top, there's a table of employees:

Role	Name	Building	Count
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: `SELECT DISTINCT building_name FROM buildings LEFT JOIN employees ON building_name = building WHERE role IS NULL;`. The results are:

Building_name
1w
2e

On the right, 'Exercise 8 — Tasks' includes two tasks:
 

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

 A green 'Continue >' button is at the bottom right.

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;

## Lesson 9

The screenshot shows the SQLBolt interface for Lesson 9. At the top, there's a table of movies:

Title	Year	Domestic Sales	International Sales	Gross Sales
The Incredibles	2004	116	8	261441092
A Bug's Life	1998	116	8	370001000

Below this, the 'Query Results' section shows the output of the query: `SELECT title, year FROM movies WHERE year % 2 = 0;`. The results are:

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

On the right, 'Exercise 9 — Tasks' includes three tasks:
 

- List all movies and their combined sales in millions of dollars ✓
- List all movies and their ratings in percent ✓
- List all movies that were released on even number years ✓

 A green 'Continue >' button is at the bottom right.

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;
```

## Lesson 10

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_employed
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 >

RESET

```
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;
```

## Lesson 11

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

Table: Employees

Role	SUM(Years_employed)
Engineer	17

Exercise 11 — Tasks

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

SELECT role, SUM(years\_employed) FROM employees GROUP BY role HAVING role = "Engineer";

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

Continue >

Next — SQL Lesson 12: Order of execution of a Query  
Previous — SQL Lesson 10: Queries with aggregates (Pt. 1)

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

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";

## Lesson 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

Exercise 12 — Tasks

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

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;

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.

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;

## Lesson 13

4 Toy Story 4 El Directore 2015 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
4	Toy Story 4	El Directore	2015	90

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.

Finish above Tasks

INSERT INTO boxoffice VALUES (4, 8.7, 340000000, 270000000);

RUN QUERY RESET

1.INSERT INTO movies VALUES (4, "Toy Story 4", "El Directore", 2015, 90);

2.INSERT INTO boxoffice VALUES (4, 8.7, 340000000, 270000000);

## Lesson 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

Exercise 14 — Tasks

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

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;

## Lesson 15

Exercise

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

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

Exercise 15 — Tasks

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

[Continue >](#)

Next — SQL Lesson 16: Creating tables  
Find SQLBolt useful? Please consider [Donating \(\\$4\) via Paypal](#) to support our site.

1.DELETE FROM movies where year < 2005;

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

## Lesson 16

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.

**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.

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

## Lesson 17

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	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
New column added		Pete Docter	2009	101	2.39	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**. ✓

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

RUN QUERY RESET

Continue >

SQL Lesson 18: Dropping tables

1.ALTER TABLE Movies ADD COLUMN Aspect\_ratio FLOAT DEFAULT 2.39;

2.ALTER TABLE Movies ADD COLUMN Language TEXT DEFAULT "English";

## Lesson 18

Query Results

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

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.

RUN QUERY RESET

Continue >

SQL Lesson 18: Dropping tables

1.DROP TABLE Movies;

## 2.DROP TABLE BoxOffice;

