

Database- Day -1: MySQL Activity SQL

Lesson 1: SELECT queries 101 SQL

This query, in particular, is really useful because it's a simple way to inspect a table by dumping all the data at once.

Exercise

We will be using a database with data about some of Pixar's classic movies for most of our exercises. This 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

```
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.

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.

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

Exercise

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

```
SELECT * FROM movies where id in (1,2,3,4,5);
```

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.

Like 3K Share % Post

2023 © SQLBolt
[Email](#) | [Twitter](#)

Additional Courses

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

AND/OR *another_condition*
AND/OR ...;

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 ✓

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

Continue >

SELECT * FROM movies where Title like "WALL-%";

RESET

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.

Like 3k Share Post

2023 © SQLBolt
[Email](#) | [Twitter](#)

Additional Courses
[Interactive Regular Expressions Lessons](#)

Lesson 4: Filtering and sorting Query results

TECHNICAL SKILLS: [Filtering and sorting](#) after introducing a new clause into the query.

Exercise

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

Id	Title	Director	Year	Length_minutes
2	Monsters University	Dan Scanlon	2013	110
11	Monsters, Inc.	Pete Docter	2001	92
14	Ratatouille	Brad Bird	2007	115
3	The Incredibles	Brad Bird	2004	116
1	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 ✓

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

Continue >

SELECT *
FROM Movies
ORDER BY Title ASC
LIMIT 5
OFFSET 5;

RESET

Next – [SQL Review: Simple SELECT Queries](#)
Previous – [SQL Lesson 3: Queries with constraints \(Pt. 2\)](#)

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

SQL Review: Simple SELECT Queries SQL

location in the world.

Did you know?
Positive latitudes correspond to the northern hemisphere, and positive longitudes correspond to the eastern hemisphere. Since North America is north of the equator and west of the prime meridian, all of the cities in the list have positive latitudes and negative longitudes.

Try and write some queries to find the information requested in the tasks you know. You may have to use a 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	Country	Population	Latitude	Longitude
Chicago	United States	2718782	41.878114	-87.629798
Houston	United States	2195914	29.760427	-95.369803

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 *  
FROM North_american_cities  
WHERE Country LIKE "United States"  
ORDER BY Population DESC  
LIMIT 2  
OFFSET 2;
```

Continue >

Next – [SQL Lesson 6: Multi-table queries with JOINS](#)
Previous – [SQL Lesson 4: Filtering and sorting Query results](#)

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

Lesson 6: Multi-table queries with JOINS SQL

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

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

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.

```
SELECT Title, Rating  
FROM Movies JOIN Boxoffice  
ON Id=Movie_id  
ORDER BY Rating DESC;
```

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.

Lesson 7: OUTER JOINS SQL

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

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 7 — Tasks

- Find the list of all buildings that have employees ✓
- Find the list of all buildings and their capacity ✓
- 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.

Lesson 8: A short note on NULLs SQL

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

Building_name	Capacity	Role	Name	Building	Years_employed
1w	32				
2e	16				

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

Exercise 8 — 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 ✓

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

Continue >

Next – [SQL Lesson 9: Queries with expressions](#)
Previous – [SQL Lesson 7: OUTER JOINS](#)

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

Lesson 9: Queries with expressions SQL

You are going to have to use expressions to transform the **boxoffice** data into something easier to understand for the tasks below.

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

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
LEFT JOIN Boxoffice ON Id=Movie_Id
WHERE Year % 2 = 0;
```

RESET

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.

Continue >

Next – [SQL Lesson 10: Queries with aggregates \(Pt. 1\)](#)

Previous – [SQL Lesson 8: A short note on NULLs](#)

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

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

```
SELECT AGG_FUNC(column_or_expression) AS aggregate_description, ...
FROM mytable
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	SUM(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 >

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.

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

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

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";

RESET

Continue >

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

Lesson 12: Order of execution of a Query SQL

Here ends our lessons on **SELECT** queries, congrats of making it this far! This exercise will try and test your understanding of queries, so don't be discouraged if you find them challenging. Just try your best.

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	55900000
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

Director	Total_Sales
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 Total_Sales
FROM Movies
LEFT JOIN Boxoffice ON Id = Movie_ID
GROUP BY Director;

RESET

Continue >

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

Lesson 13: Inserting rows SQL

columns defined.

Since the following lessons will modify the database, you'll have to manually run each query once they are ready to go.

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	John Lasseter	2017	123

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

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 >

Lesson 14: Updating rows SQL

row that satisfies the constraint in the **WHERE** clause.

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

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

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 >

Lesson 15: Deleting rows SQL

```
WHERE condition;
```

If you decide to leave out the **WHERE** constraint, then *all* rows are removed, which is a quick and easy 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

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

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. ✓

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

Continue >

RUN QUERY RESET

Lesson 16: Creating tables SQL

```
Movies table schema  
  
CREATE TABLE movies (  
  id INTEGER PRIMARY KEY,  
  title TEXT,  
  director TEXT,  
  year INTEGER,  
  length_minutes INTEGER  
);
```

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 downloadedThis table has no constraints. ✓

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

Continue >

RUN QUERY RESET

Next – SQL Lesson 17: Altering tables

Find SQLBolt useful? Please consider

Lesson 17: Altering tables SQL

If you need to rename the table itself, you can also do that using the **RENAME TO** clause of the statement.

```
Altering table name
ALTER TABLE mytable
RENAME TO new_table_name;
```

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

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

Lesson 18: Dropping tables SQL

Exercise

We've reached the end of our exercises, so lets clean up by removing all the tables we've worked with.

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
----	-------	----------	------	----------------

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 >

 **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.

Continue to More Topics >