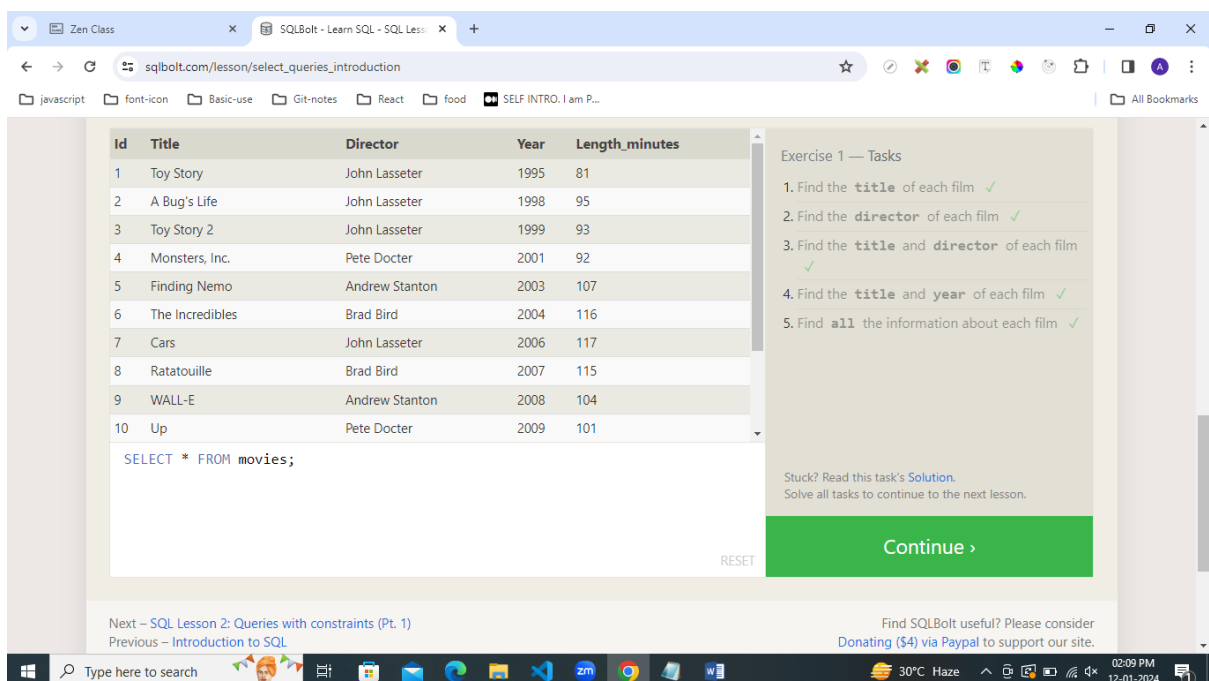


My Sql Task -1

1. 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



The screenshot shows the SQLBolt website interface. On the left, there is a table of Pixar movies with columns: Id, Title, Director, Year, and Length_minutes. Below the table is a text area containing the SQL query `SELECT * FROM movies;` and a 'RESET' button. On the right, there is a section titled 'Exercise 1 — Tasks' with five tasks listed. A green 'Continue >' button is at the bottom right of the exercise section. At the very bottom of the page, there are navigation links for 'Next - SQL Lesson 2: Queries with constraints (Pt. 1)' and 'Previous - Introduction to SQL', along with a donation prompt.

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 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.

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

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

Table: Movies

Title	Year
Toy Story	1995
A Bug's Life	1998
Toy Story 2	1999
Monsters, Inc.	2001
Finding Nemo	2003

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.

`Select title, year from movies Where year <= 2003;`

RESET Continue >

3. Here's the definition of a query with a WHERE clause again, go ahead and try and write some queries with the operators above to limit the results to the information we need in the tasks below.

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.

`SELECT * from movies where title like "wall-*";`

RESET Continue >

4. There are a few concepts in this lesson, but all are pretty straightforward 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.

The screenshot shows the SQLBolt web application interface. At the top, the browser tabs show 'Zen Class' and 'SQLBolt - Learn SQL - SQL Lesson'. The address bar shows 'sqlbolt.com/lesson/filtering_sorting_query_results'. Below the browser, there's a navigation bar with links like 'javascript', 'font-icon', 'Basic-use', 'Git-notes', 'React', 'food', and 'SELF INTRO. I am P...'. The main content area is divided into two parts. On the left, under the heading 'Table: Movies', there's a table with the following data:

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

Below the table, there's a text input field containing the SQL query: `SELECT title FROM movies order by title asc limit 5 offset 5;`. To the right of the table, there's a section titled 'Exercise 4 — Tasks' with four tasks, each marked with a green checkmark:

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, there's a link 'Stuck? Read this task's Solution.' and a button 'Continue >'. At the bottom of the screen, there's a Windows taskbar with various icons and a system tray showing '26°C Partly cloudy' and '07:47 PM 12-01-2024'.

5. 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	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 >

6. We've added a new table to the Pixar database so that you can try practicing some joins. The **BoxOffice** table stores information about the ratings and sales of each particular Pixar movie, and the **Movie_id** column in that table corresponds with the **Id** column in the **Movies** table 1-to-1. Try and solve the tasks below using the **INNER JOIN** introduced above

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 title, rating FROM movies join boxoffice on movies.id=boxoffice .movie_id order by 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.

Continue >

7. In this exercise, you are going to be working with a new table which stores fictional data about **Employees** in the film studio and their assigned office **Buildings**. Some of the buildings are new, so they don't have any employees in them yet, but we need to find some information about them regardless.

Since our browser SQL database is somewhat limited, only the **LEFT JOIN** is supported in the exercise below.

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

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 >

8. 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.

Query Results

Building_name
1w
2e

No such table: employee

```
SELECT distinct building_name FROM buildings left join employee on
building_name = building WHERE role 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 >](#)

RESET

https://sqlbolt.com/lesson/select_queries_with_expressions

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

Exercise 9 — 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 ✓

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

[Continue >](#)

RESET

```
SELECT title, year FROM movies WHERE year % 2 = 0;
```

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.

10. 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.

The screenshot shows a web browser window with the URL `sqlbolt.com/lesson/select_queries_with_aggregates`. The page displays a table titled "Table: Employees" with the following data:

Building	Total_years_employed
1e	29
2w	36

Below the table, the SQL query is shown:

```
SELECT building, Sum(years_employed) as Total_years_employed
FROM employees
group by building;
```

To the right of the table, there is a section titled "Exercise 10 — Tasks" with three 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 ✓

Below the tasks, there is a link to the solution: "Stuck? Read this task's Solution. Solve all tasks to continue to the next lesson." and a green "Continue >" button.

11. 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.

The screenshot shows a web browser window with the URL `sqlbolt.com/lesson/select_queries_with_aggregates_pt_2`. The page displays a table titled "Table: Employees" with the following data:

Role	Sum(Years_employed)
Engineer	17

Below the table, the SQL query is shown:

```
SELECT role, sum(years_employed) FROM employees
group by role
having role = "Engineer";
```

To the right of the table, there is a section titled "Exercise 11 — Tasks" with three 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 ✓

Below the tasks, there is a link to the solution: "Stuck? Read this task's Solution. Solve all tasks to continue to the next lesson." and a green "Continue >" button.

12. 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.

The screenshot shows the SQLBolt website interface for Lesson 12. On the left, a table displays the cumulative sales for various directors. In the center, a SQL query is provided to calculate these sales. On the right, two tasks are listed for the exercise. The interface includes a 'Continue' button and a 'RESET' link.

Director	Cumulative_sales_from_all_movies
Andrew Stanton	1458055121
Brad Bird	1255164910
Brenda Chapman	538983207
Dan Scanlon	743559607
John Lasseter	2232208025
Lee Unkrich	1063171911
Pete Docter	1294159000

```
SELECT director, sum(domestic_sales + international_sales) as  
Cumulative_sales_from_all_movies FROM movies inner join boxoffice on  
movies.id = boxoffice.movie_id group by director;
```

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.

13. In this exercise, we are going to play studio executive and add a few movies to the **Movies** to our portfolio. In this table, the **Id** is an auto-incrementing integer, so you can try inserting a row with only the other columns defined.

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

SQLBolt - Learn SQL - SQL Lesson 13

sqlbolt.com/lesson/inserting_rows

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.

Continue >

RUN QUERY RESET

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

Zen Class SQLBolt - Learn SQL - SQL Lesson 14

sqlbolt.com/lesson/updating_rows

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.

Continue >

RUN QUERY RESET

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

Browser tabs: Zen Class, SQLBolt - Learn SQL - SQL Les, translate - Google Search

Address bar: sqlbolt.com/lesson/deleting_rows

Navigation: javascript, font-icon, Basic-use, Git-notes, React, food, DB, All Bookmarks

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 >](#)

Buttons: RUN QUERY, RESET

Windows taskbar: Type here to search, 07:45 PM, 16-01-2024

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

Browser tabs: SQLBolt - Learn SQL - SQL Les, +

Address bar: sqlbolt.com/lesson/creating_tables

Navigation: javascript, font-icon, Basic-use, Git-notes, React, food, DB, All Bookmarks

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
MongoDB	6.2.0	265000000

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 >](#)

Code input: `select * from Database;`

Windows taskbar: Type here to search, 24°C Mostly cloudy, 10:07 PM, 16-01-2024

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

Table: Movies

3	Toy Story 2	John Lasseter	1999	93	2.39	English
4	Monsters, Inc.	Pete Docter	2001	92	2.39	English
5	Finding Nemo	Andrew Stanton	2003	107	2.39	English
6	The Incredibles	Brad Bird	2004	116	2.39	English
7	Cars	John Lasseter	2006	117	2.39	English
8	Ratatouille	Brad Bird	2007	115	2.39	English
9	WALL-E	Andrew Stanton	2008	104	2.39	English
10	Up	Pete Docter	2009	101	2.39	English
11	Toy Story 3	Lee Unkrich	2010	103	2.39	English
12	Cars 2	John Lasseter	2011	120	2.39	English
13	Brave	Brenda Chapman	2012	102	2.39	English

```
ALTER TABLE Movies
ADD COLUMN Language FLOAT DEFAULT English;
```

Exercise 17 — Tasks

1. Add a column named **Aspect_ratio** with a **FLOAT** data type to store the aspect-ratio each movie was released in. ✓
2. Add another column named **Language** with a **TEXT** data type to store the language that the movie was released in. Ensure that the default for this language is **English**. ✓

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

Continue >

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

Query Results

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

```
DROP TABLE BoxOffice;
```

Exercise 18 — Tasks

1. We've sadly reached the end of our lessons, let's 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 >