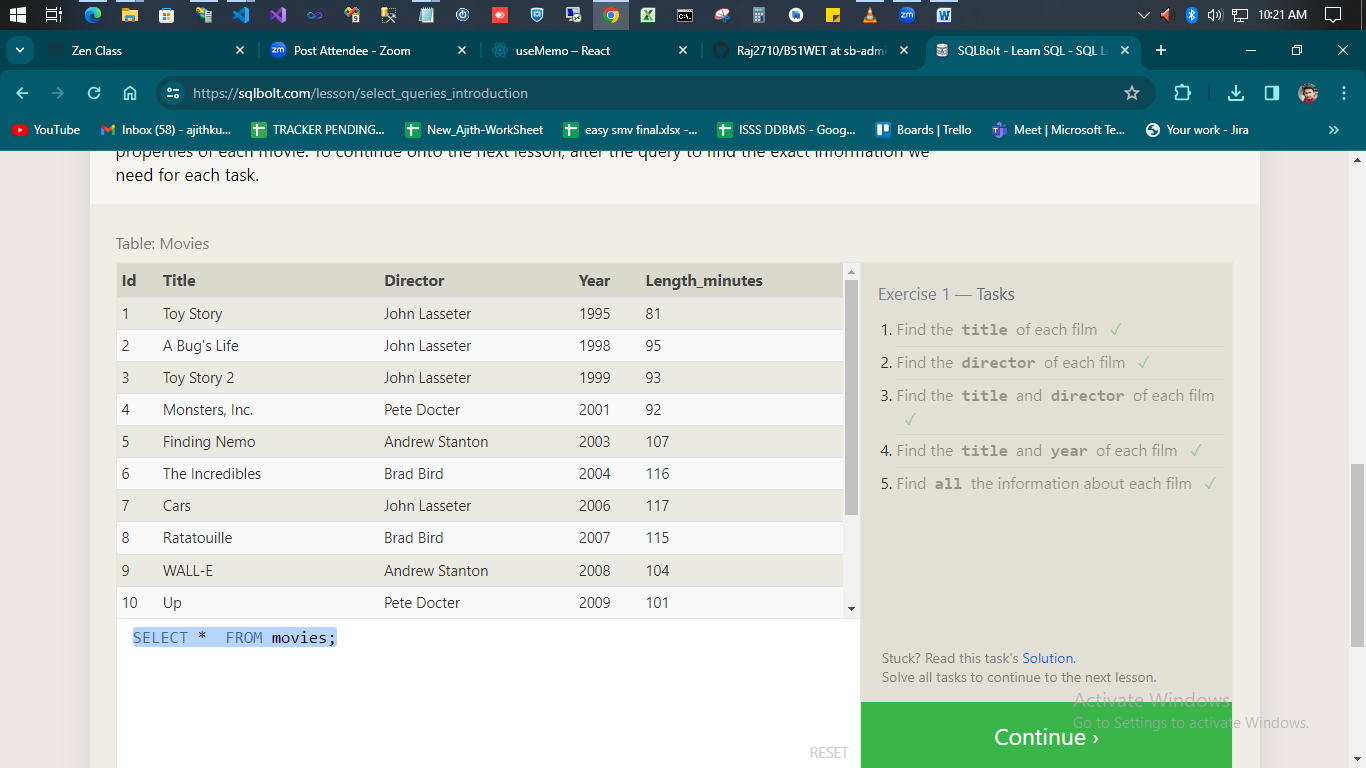
**SQL Lesson 1: SELECT queries 101**



1. **Find the title of each film**

SELECT title FROM movies;

**2.Find the director of each film**

SELECT director FROM movies;

**3.Find the title and director of each film**

SELECT title , director FROM movies;

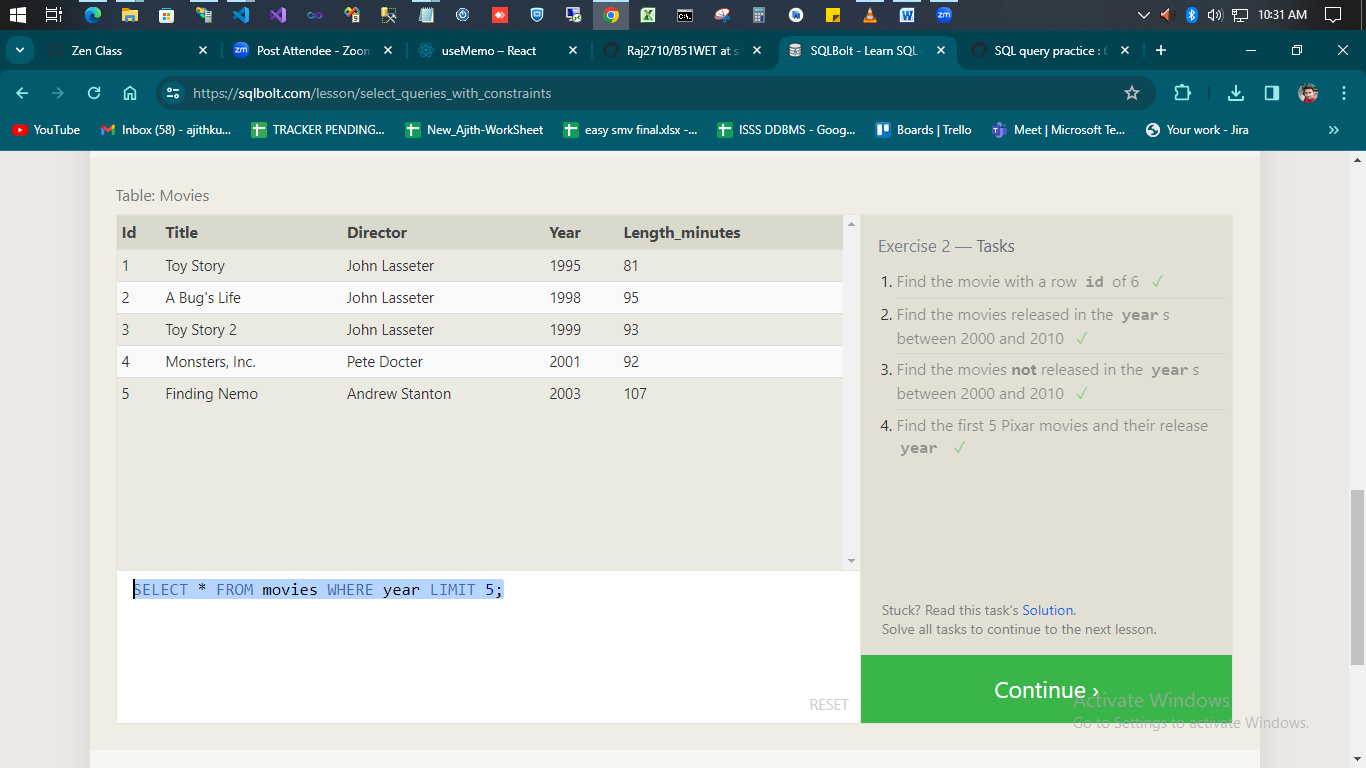
1. **Find the title and year of each film**

SELECT title , year FROM movies;

**5.Find all the information about each film**

SELECT \* FROM movies;

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



1. **Find the movie with a row id of 6**

SELECT \* FROM movies where id=6

1. **Find the movies released in the years between 2000 and 2010**

SELECT \* FROM movies where year between 2000 and 2010

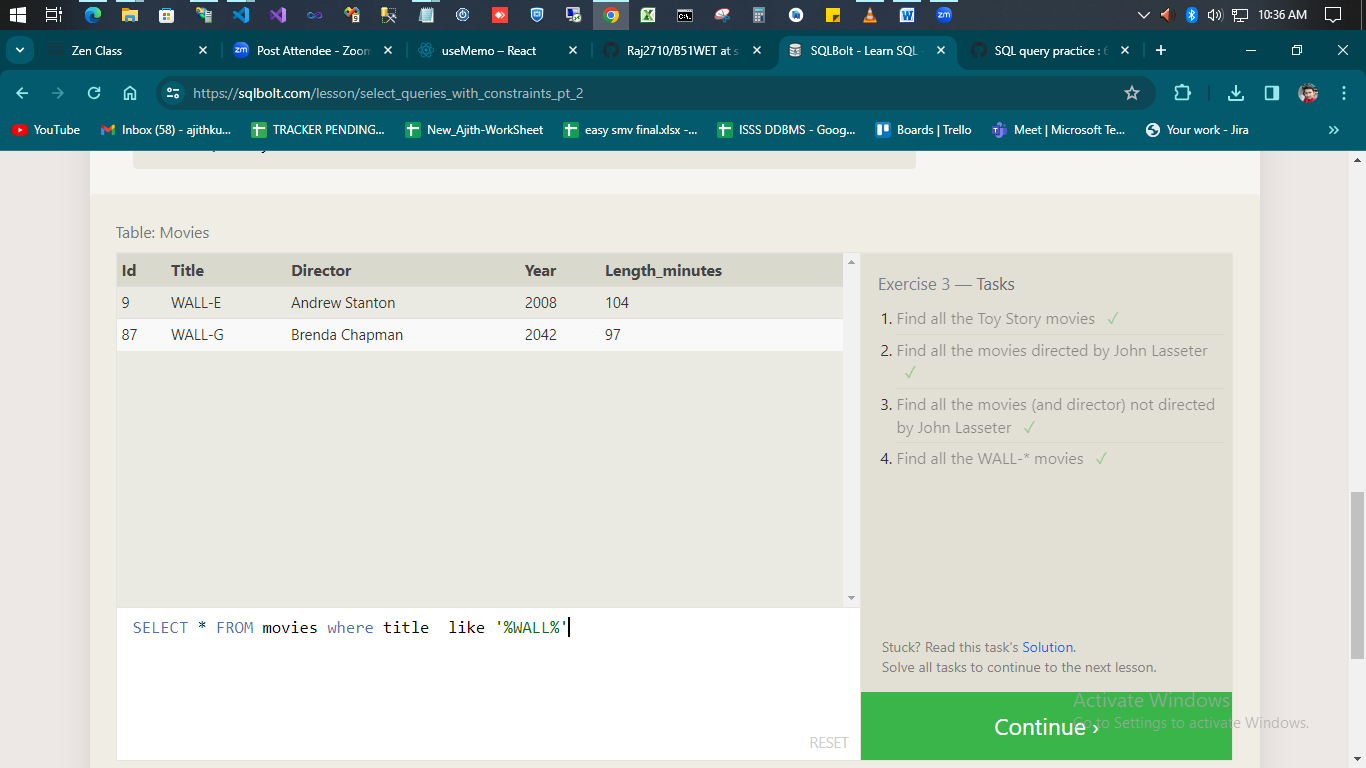
1. **Find the movies not released in the years between 2000 and 2010**

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

1. **Find the first 5 Pixar movies and their release year**

SELECT \* FROM movies WHERE year LIMIT 5;

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



1. **Find all the Toy Story movies**

SELECT \* FROM movies where title like '%Toy%'

1. **Find all the movies directed by John Lasseter**

SELECT \* FROM movies where director like '%Lasseter%'

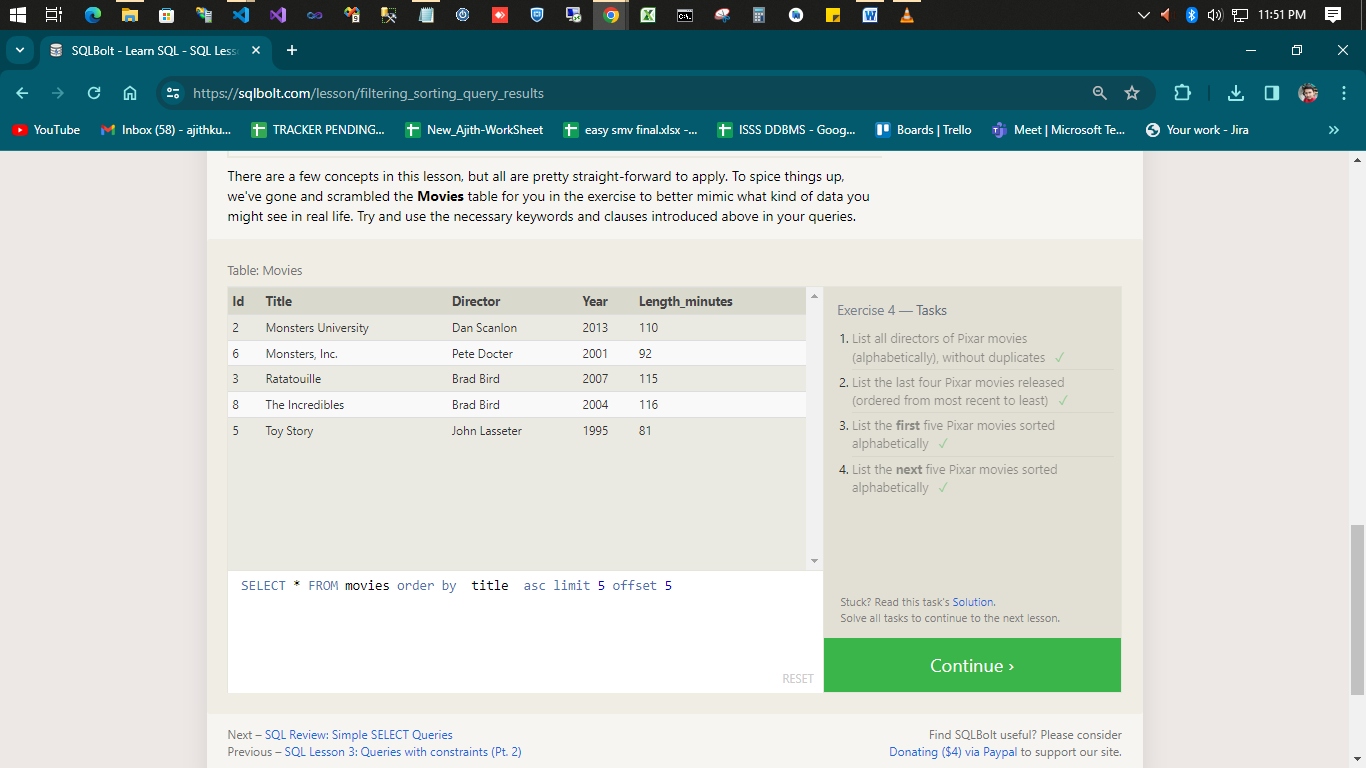
1. **Find all the movies (and director) not directed by John Lasseter**

SELECT \* FROM movies where director not like '%Lasseter%'

1. **Find all the WALL-\* movies**

SELECT \* FROM movies where title like '%WALL%'

**SQL Lesson 4: Filtering and sorting Query results**



1. **List all directors of Pixar movies (alphabetically), without duplicates**

SELECT distinct director FROM movies order by director asc

1. **List the last four Pixar movies released (ordered from most recent to least)**

SELECT \* FROM movies order by year desc limit 4

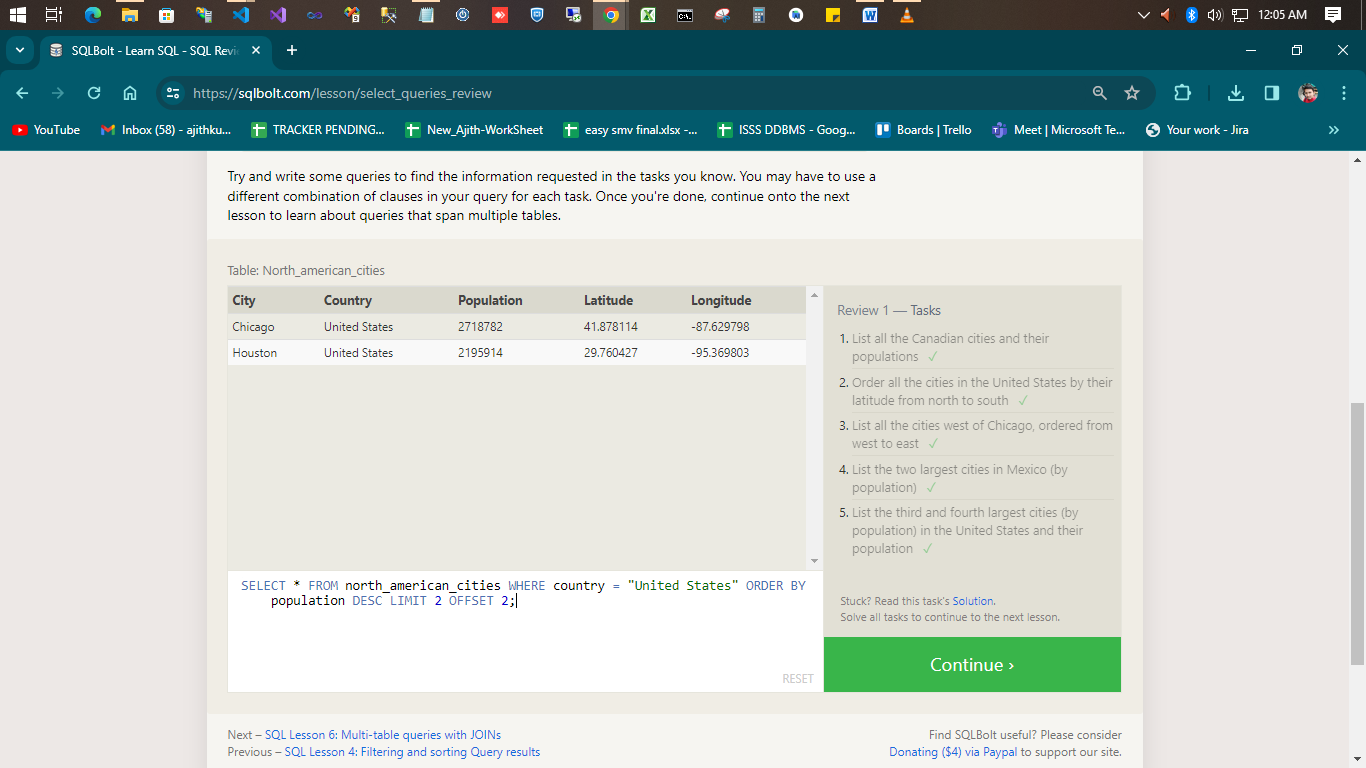
1. **List the first five Pixar movies sorted alphabetically**

SELECT \* FROM movies order by title asc limit 5

1. **List the next five Pixar movies sorted alphabetically**

SELECT \* FROM movies order by title asc limit 5 offset 5

**SQL Review: Simple SELECT Queries**



1. **List all the Canadian cities and their populations**

SELECT \* FROM north\_american\_cities where country='Canada'

1. **Order all the cities in the United States by their latitude from north to south**

SELECT \* FROM north\_american\_cities where country='United States' order by Latitude desc

1. **List all the cities west of Chicago, ordered from west to east**

SELECT \* FROM north\_american\_cities where Longitude < -87.629798 order by Longitude asc

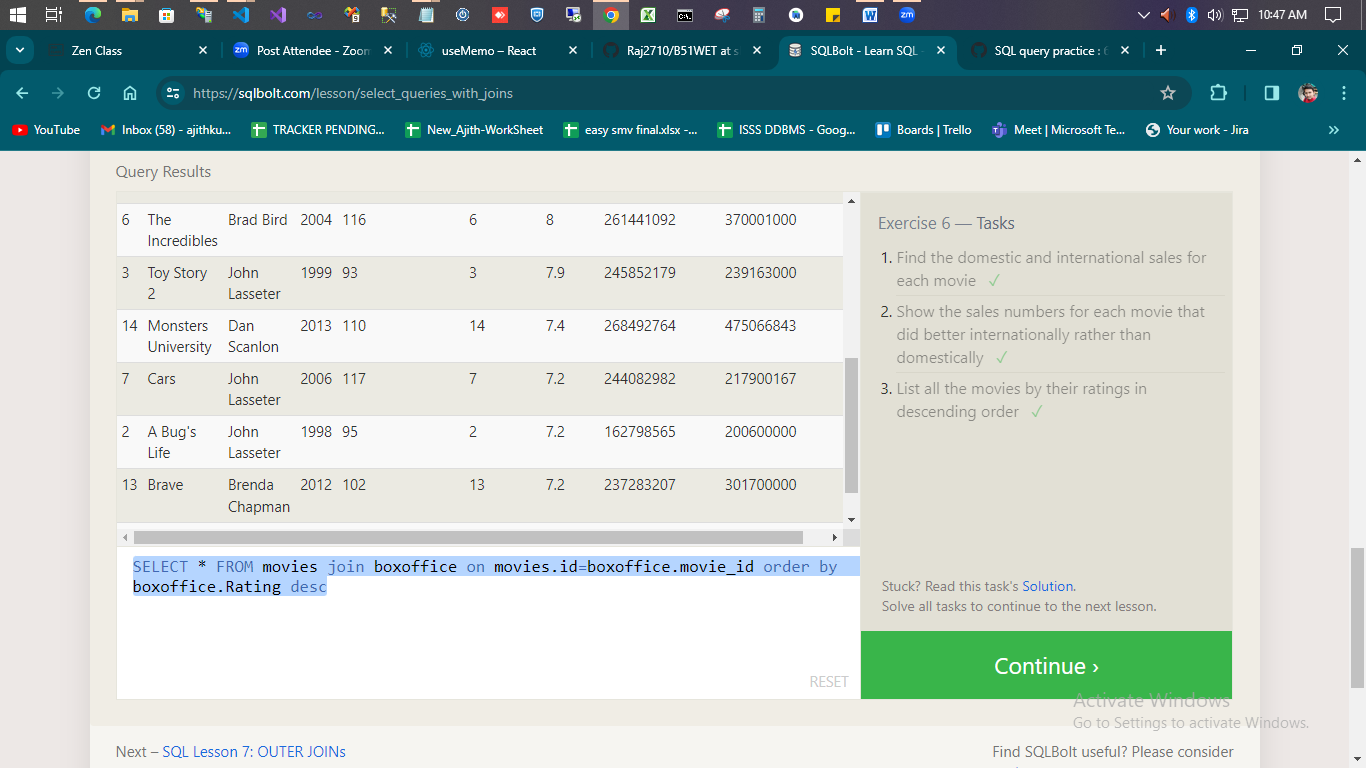
1. **List the two largest cities in Mexico (by population)**

SELECT \* FROM north\_american\_cities WHERE country = ‘Mexico’ ORDER BY population DESC LIMIT 2;

1. **List the third and fourth largest cities (by population) in the United States and their population**

SELECT \* FROM north\_american\_cities WHERE country = ‘United States’ ORDER BY population DESC LIMIT 2 OFFSET 2;

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



1. **Find the domestic and international sales for each movie**

SELECT \* FROM movies join boxoffice on movies.id=boxoffice.movie\_id

1. **Show the sales numbers for each movie that did better internationally rather than domestically**

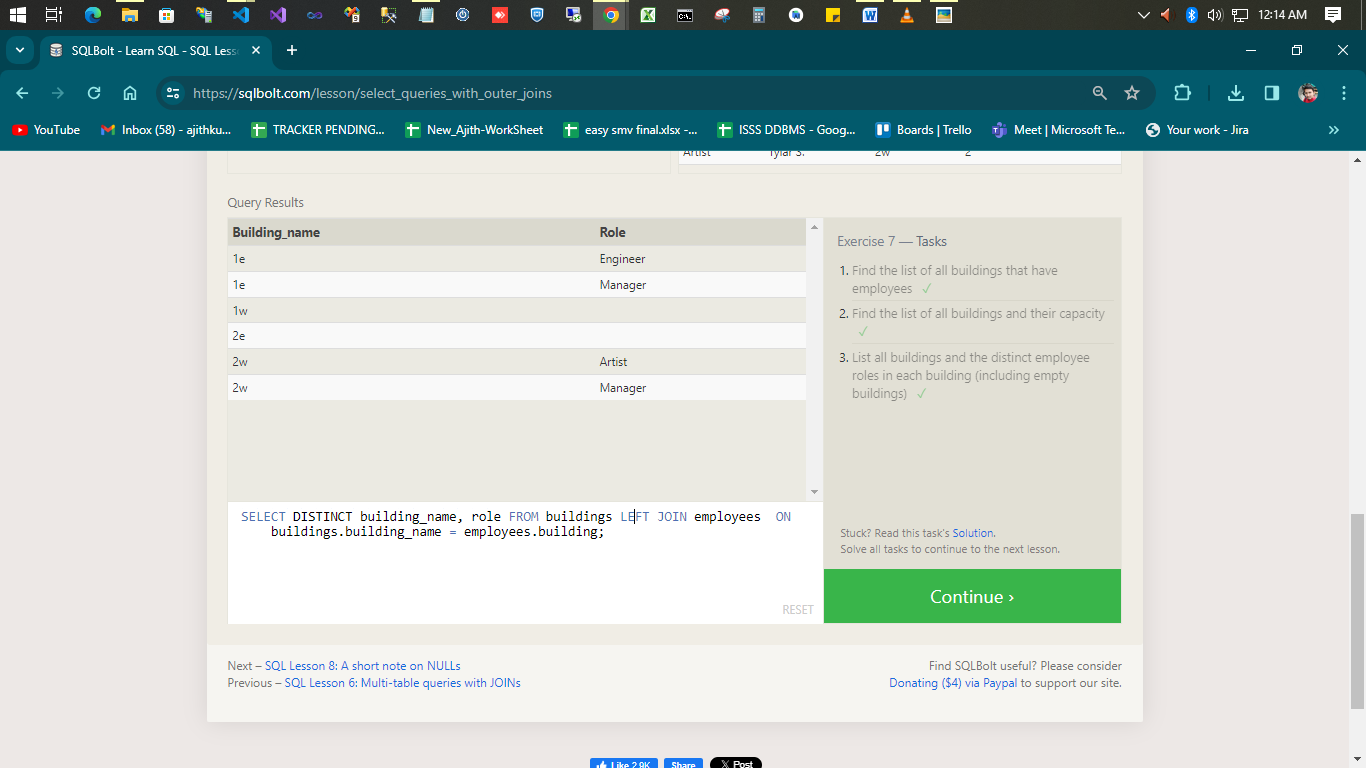
SELECT \* FROM movies join boxoffice on movies.id=boxoffice.movie\_id where boxoffice.International\_sales >=boxoffice.Domestic\_sales

1. **List all the movies by their ratings in descending order**

SELECT \* FROM movies join boxoffice on movies.id=boxoffice.movie\_id order by

boxoffice.Rating desc

**SQL Lesson 7: OUTER JOINs**



1. **Find the list of all buildings that have employees**

SELECT distinct Building FROM Employees ;

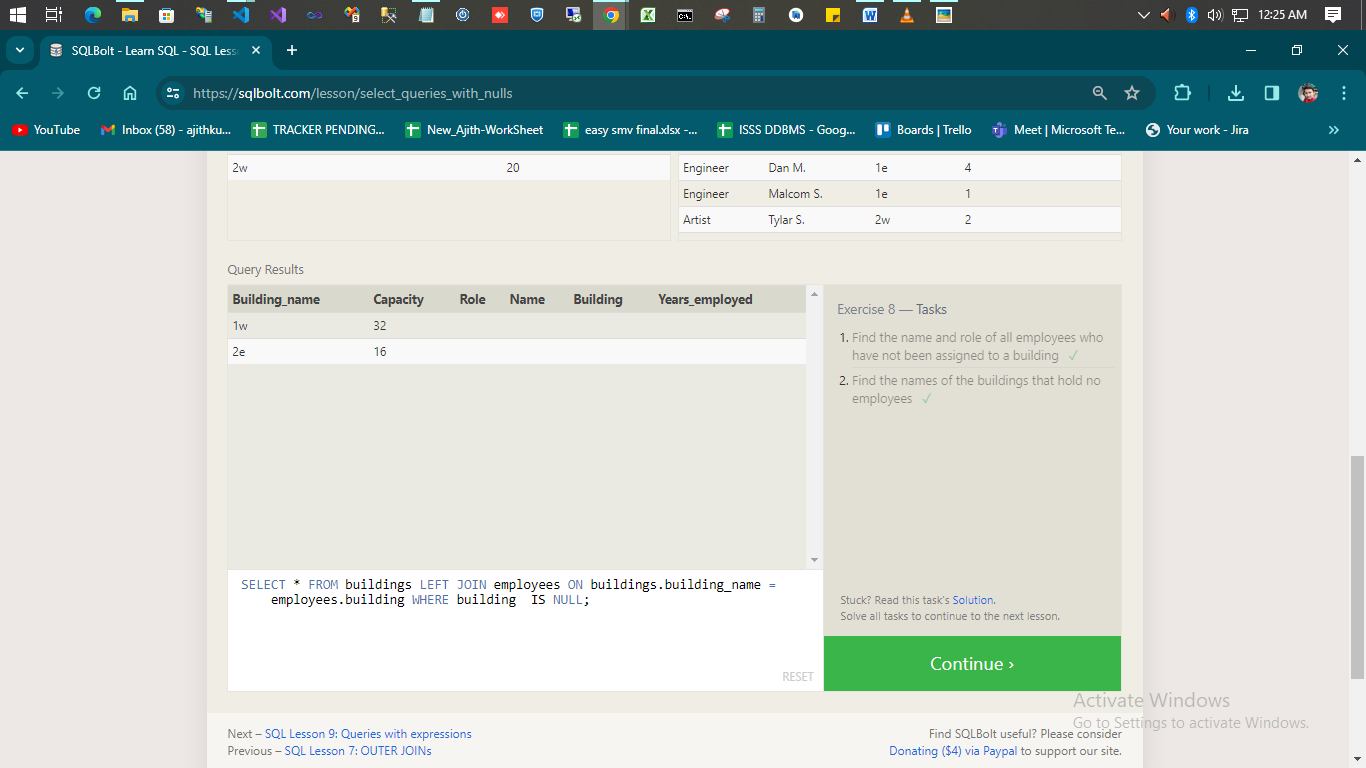
1. **Find the list of all buildings and their capacity**

SELECT \* FROM Buildings

1. **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;

**SQL Lesson 8: A short note on NULLs**



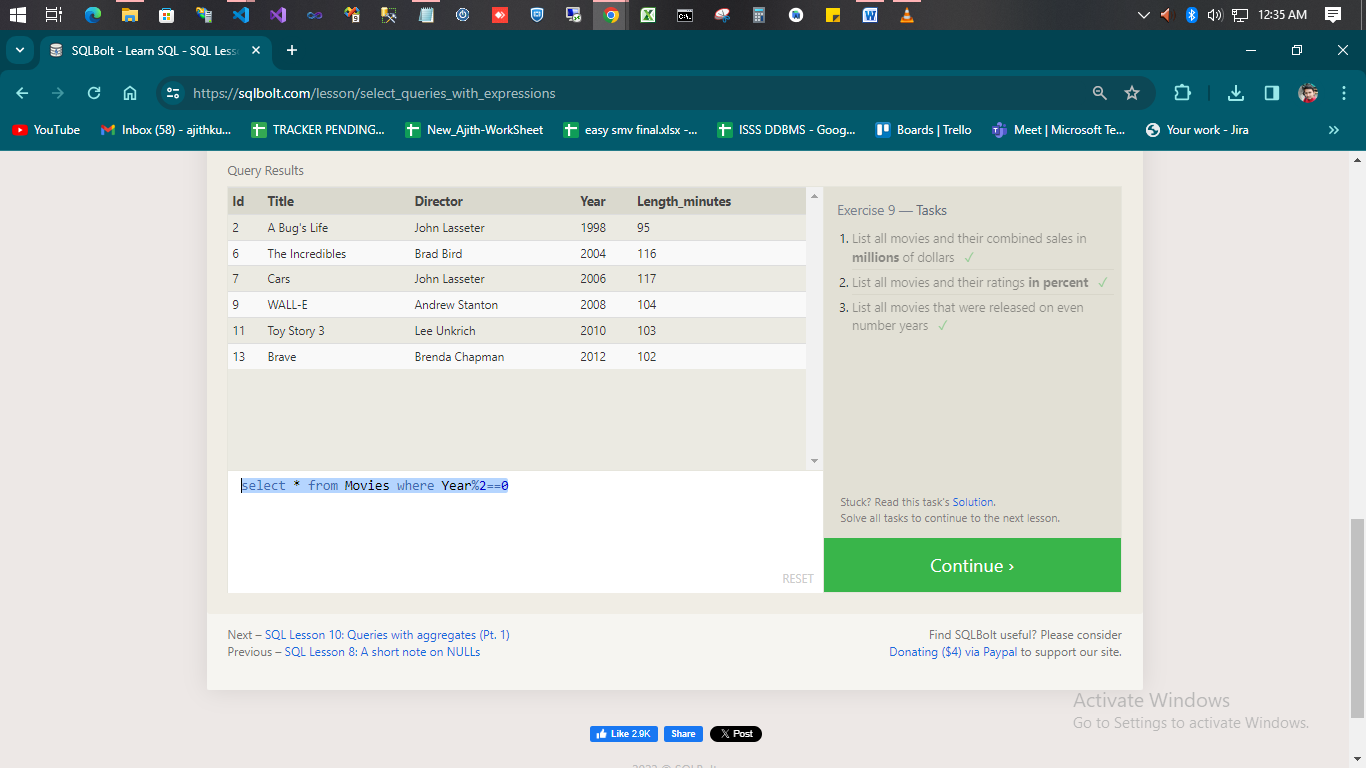
1. **Find the name and role of all employees who have not been assigned to a building**

SELECT \* FROM employees WHERE buildings IS NULL;

1. **Find the names of the buildings that hold no employees**

SELECT \* FROM buildings LEFT JOIN employees ON buildings.building\_name = employees.building WHERE building IS NULL;

**SQL Lesson 9: Queries with expressions**



1. **List all movies and their combined sales in millions of dollars**

select id,title ,(Domestic\_sales+International\_sales)/1000000 as sales from movies join boxoffice on id=movie\_id

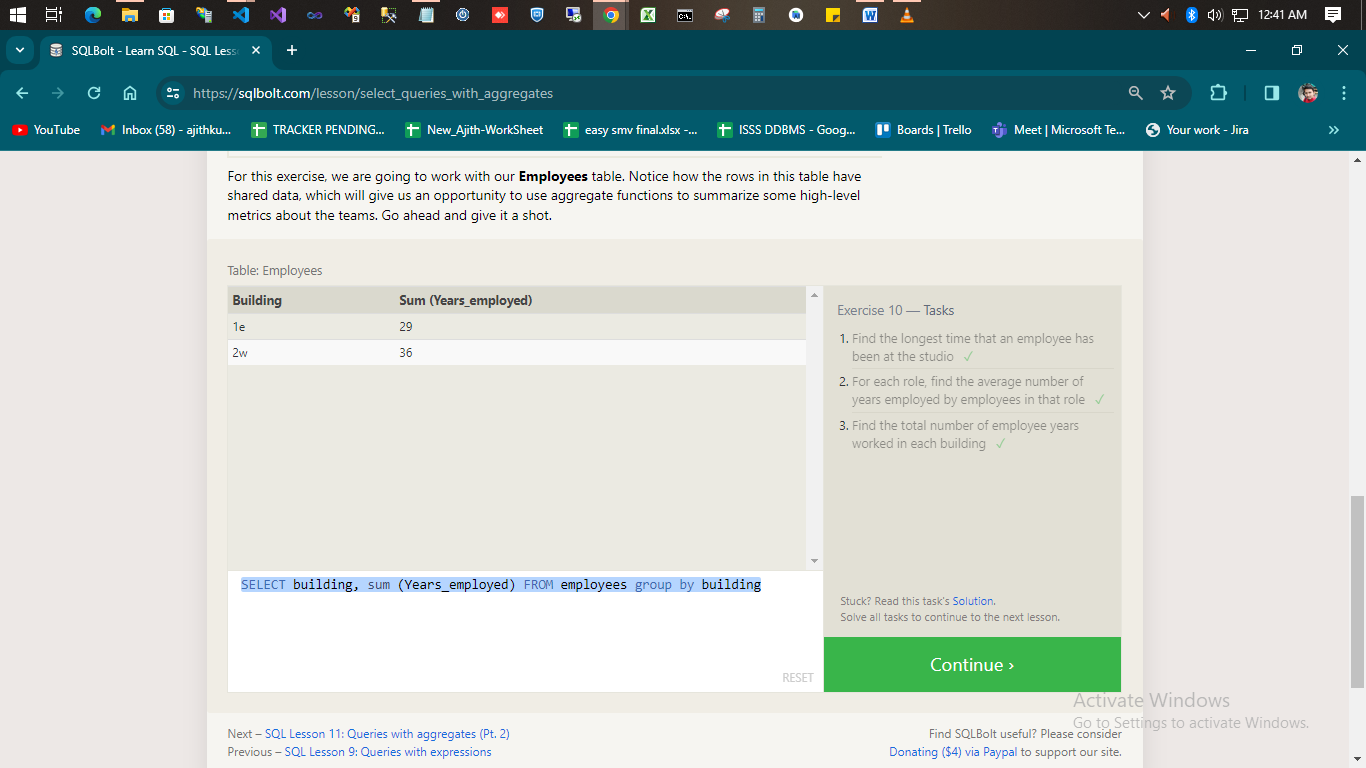
1. **List all movies and their ratings in percent**

SELECT title, rating\*10 AS ratings\_percent FROM Movies JOIN Boxoffice ON Movies.id = Boxoffice.movie\_id;

1. **List all movies that were released on even number years**

select \* from Movies where Year%2==0

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



1. **Find the longest time that an employee has been at the studio**

SELECT max(Years\_employed) FROM employees;

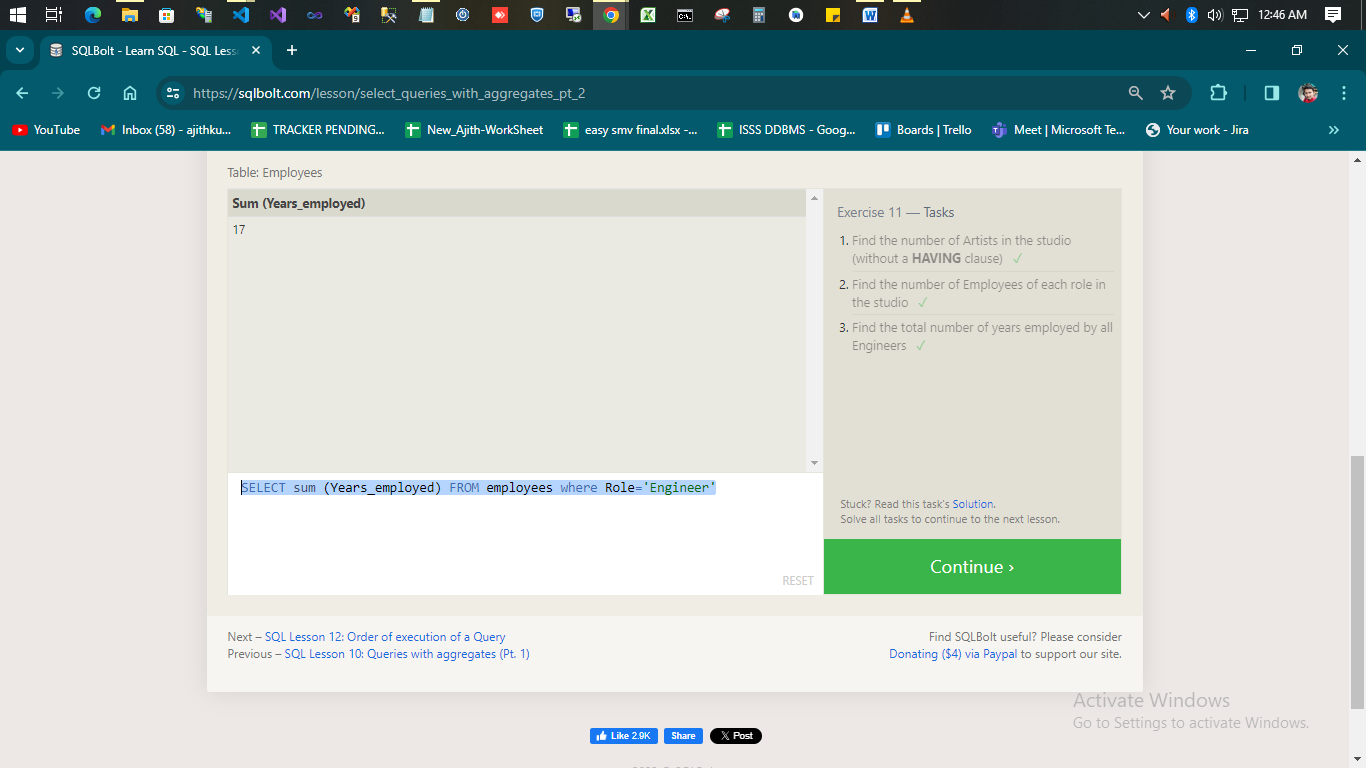
1. **For each role, find the average number of years employed by employees in that role**

SELECT role, avg(Years\_employed) FROM employees group by role

1. **Find the total number of employee years worked in each building**

SELECT building, sum (Years\_employed) FROM employees group by building

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



1. **Find the number of Artists in the studio (without a HAVING clause)**

SELECT count(\*) FROM employees where Role='Artist'

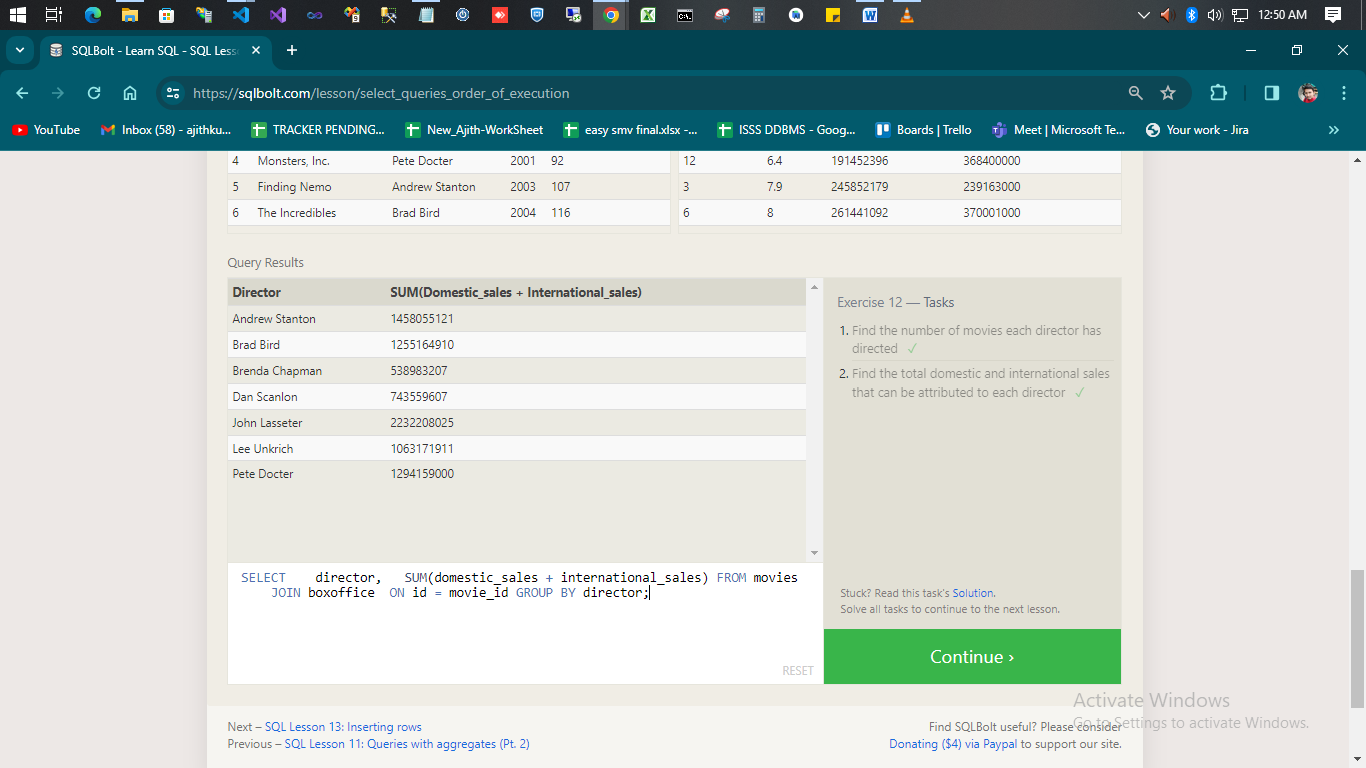
1. **Find the number of Employees of each role in the studio**

SELECT role,count(\*) FROM employees group by Role

1. **Find the total number of years employed by all Engineers**

SELECT sum (Years\_employed) FROM employees where Role='Engineer'

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



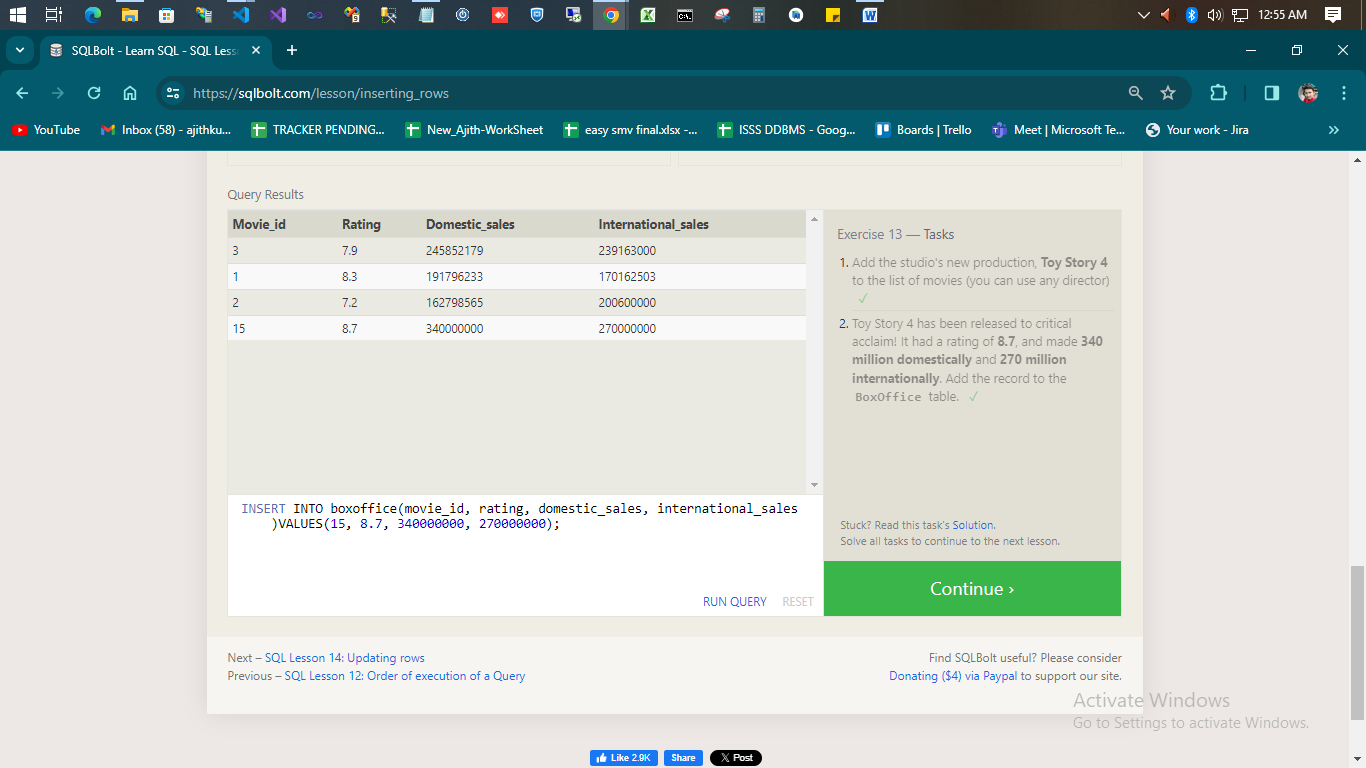
1. **Find the number of movies each director has directed**

SELECT director,COUNT(\*) FROM movies GROUP BY director;

1. **Find the total domestic and international sales that can be attributed to each director**

SELECT director, SUM(domestic\_sales + international\_sales) FROM movies JOIN boxoffice ON id = movie\_id GROUP BY director;

**SQL Lesson 13: Inserting rows**



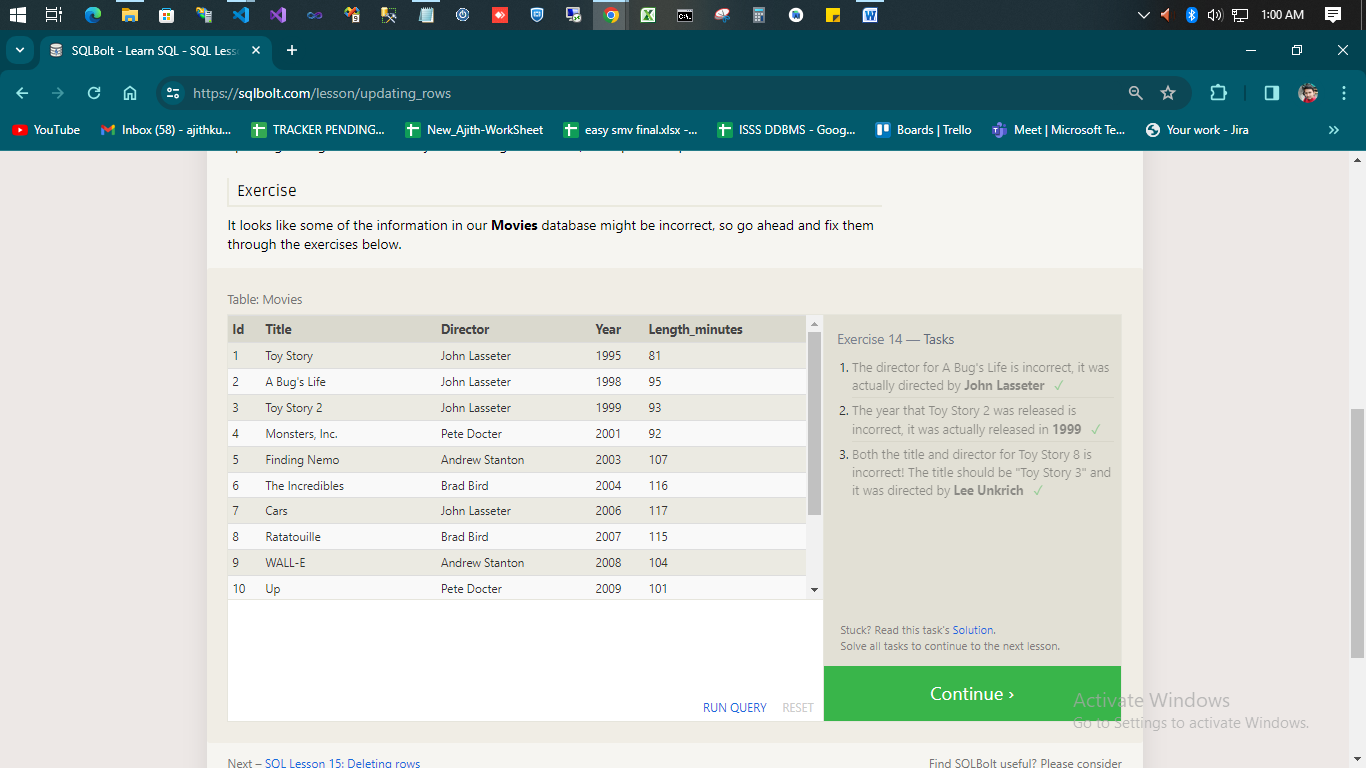
1. **Add the studio's new production, Toy Story 4 to the list of movies (you can use any director)**

INSERT INTO movies (title, director, year, length\_minutes)VALUES ('Toy Story 4', 'Lance Lafontaine', 2984, 15);

1. **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);

**SQL Lesson 14: Updating rows**



1. **The director for A Bug's Life is incorrect, it was actually directed by John Lasseter**

UPDATE movies SET director = ‘John Lasseter’ WHERE title = ‘A Bug's Life’;

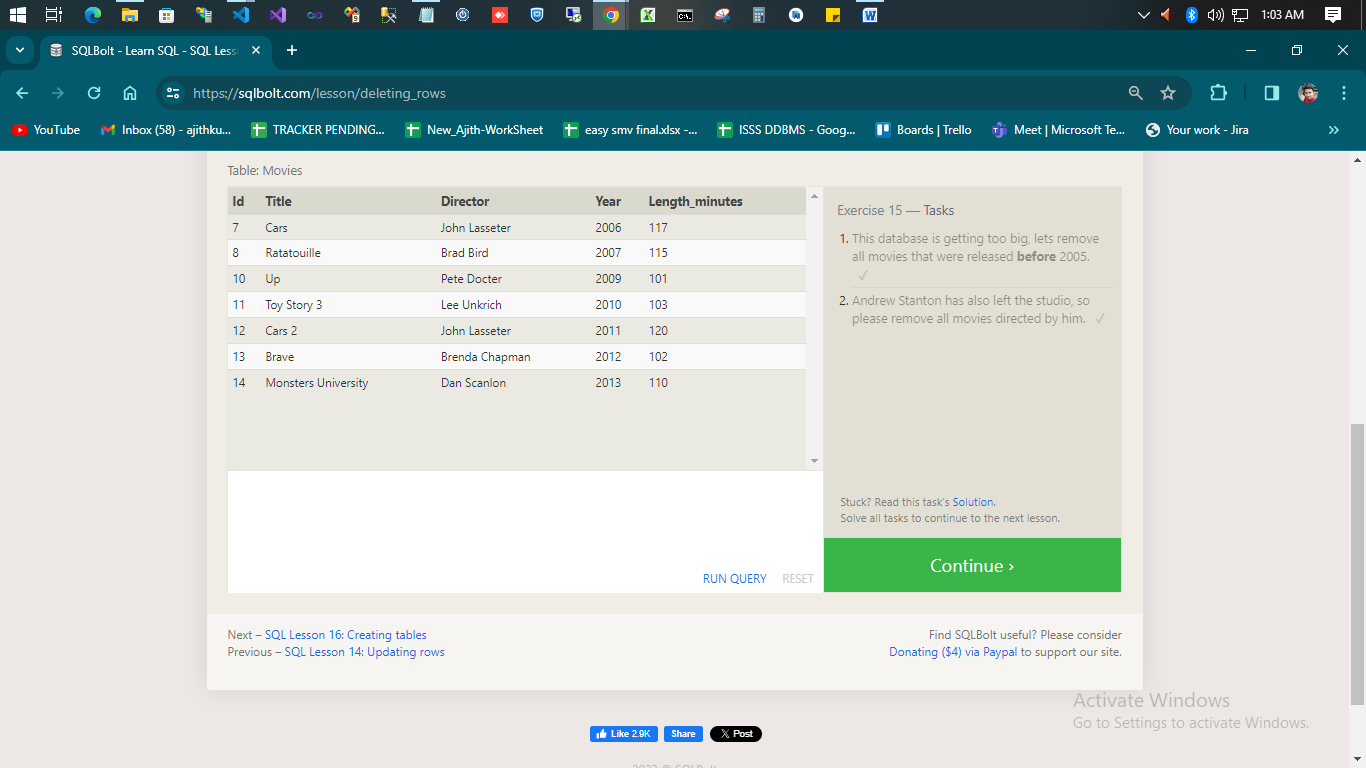
1. **The year that Toy Story 2 was released is incorrect, it was actually released in 1999**

Update Movies set Year = 1999 where Title = 'Toy Story 2';

1. **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**

Update Movies set Title = 'Toy Story 3', Director = 'Lee Unkrich' where Title = 'Toy Story 8';

**SQL Lesson 15: Deleting rows**



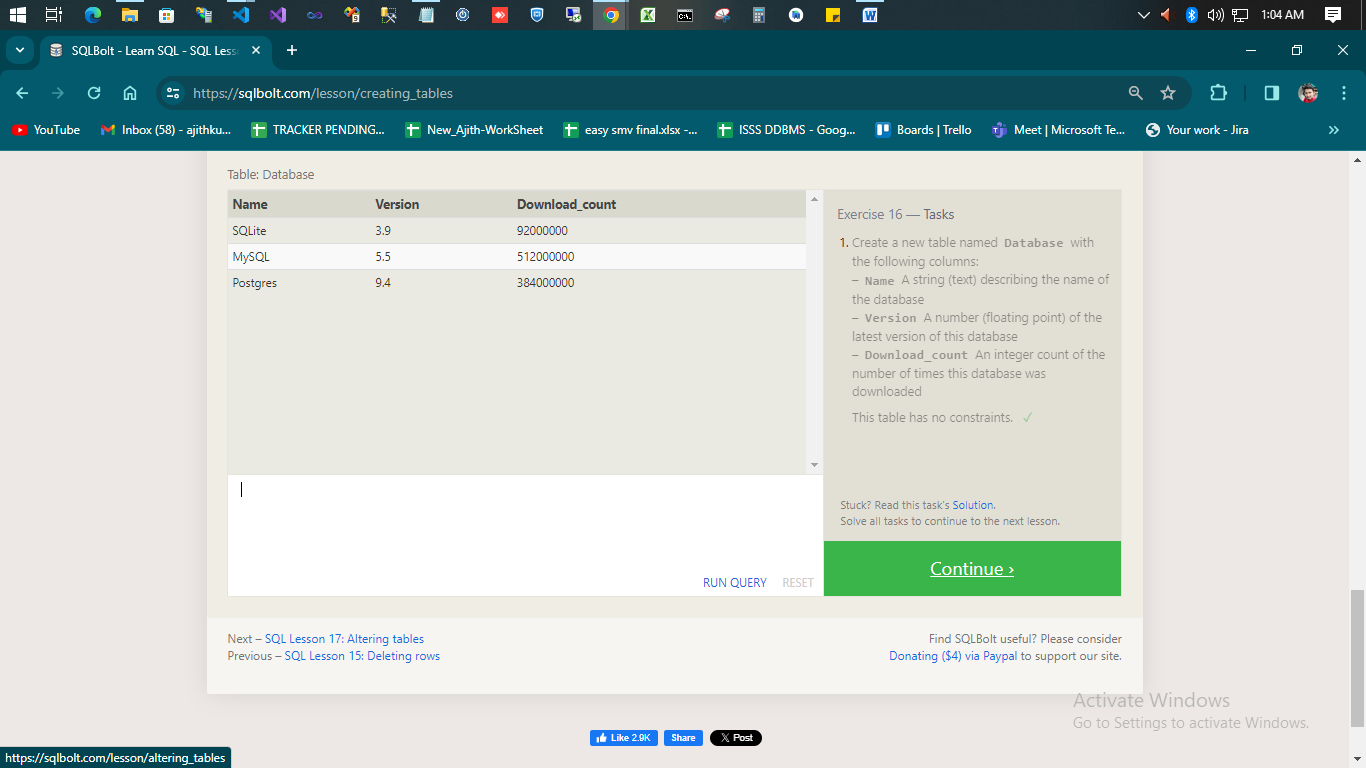
1. **This database is getting too big, lets remove all movies that were released before 2005.**

Delete from movies where Year < 2005;

1. **Andrew Stanton has also left the studio, so please remove all movies directed by him.**

Delete from movies where Director = 'Andrew Stanton';

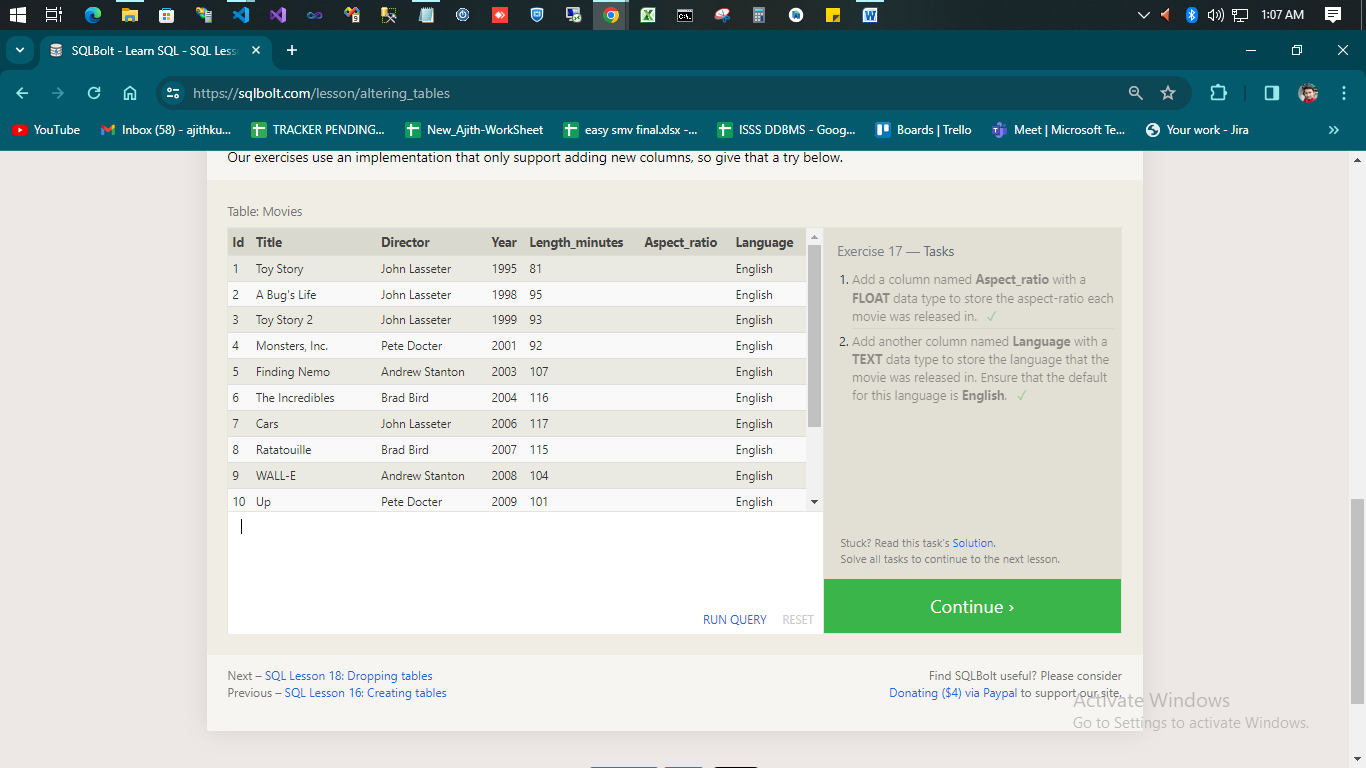
**SQL Lesson 16: Creating tables**



1. **Create a new table named Database with the following columns: – Name A string (text) describing the name of the database**

Create table Database(Name Varchar(20), Version FLOAT, Download\_count int);

**SQL Lesson 17: Altering tables**



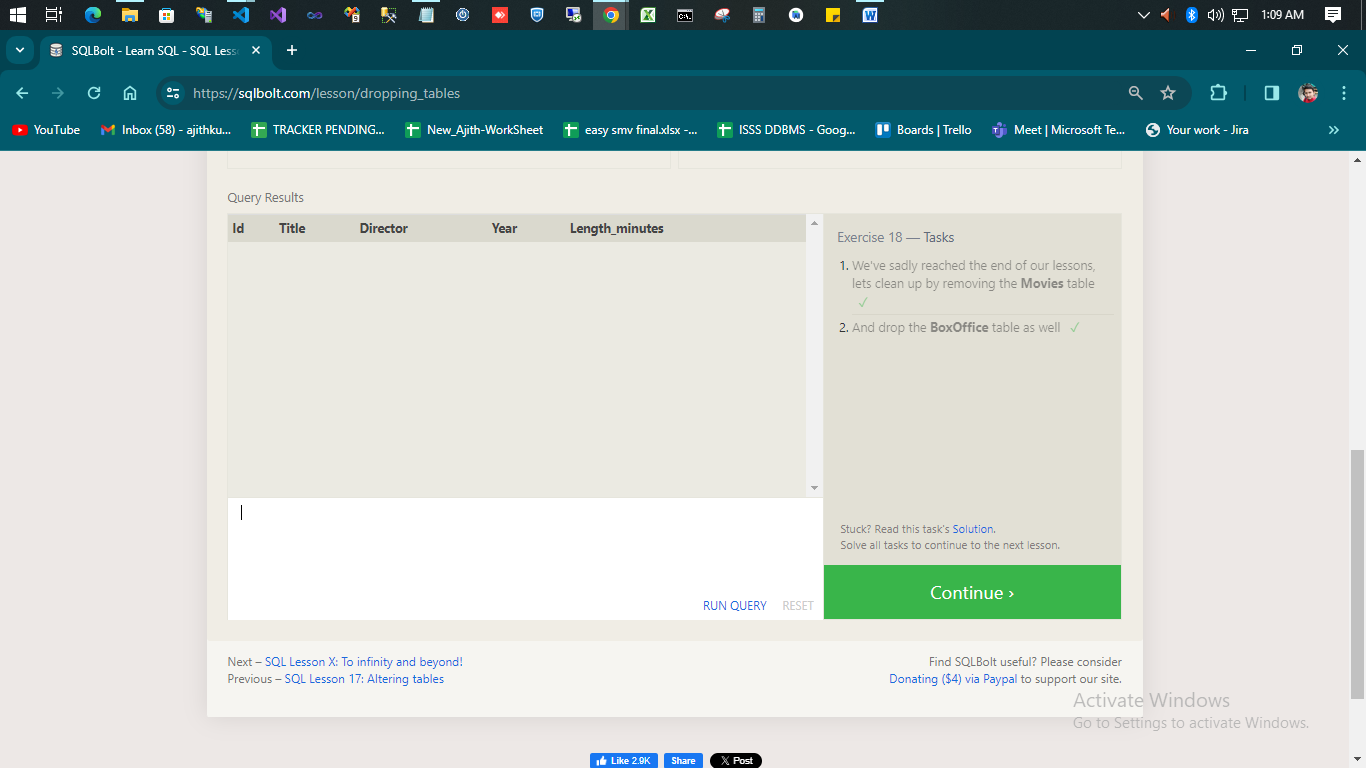
1. **Add a column named Aspect\_ratio with a FLOAT data type to store the aspect-ratio each movie was released in.**

Alter table Movies add Aspect\_ratio float;

1. **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';

**SQL Lesson 18: Dropping tables**



1. **We've sadly reached the end of our lessons, lets clean up by removing the Movies table;**

Drop table Movies

1. **And drop the BoxOffice table as well**

Drop table BoxOffice