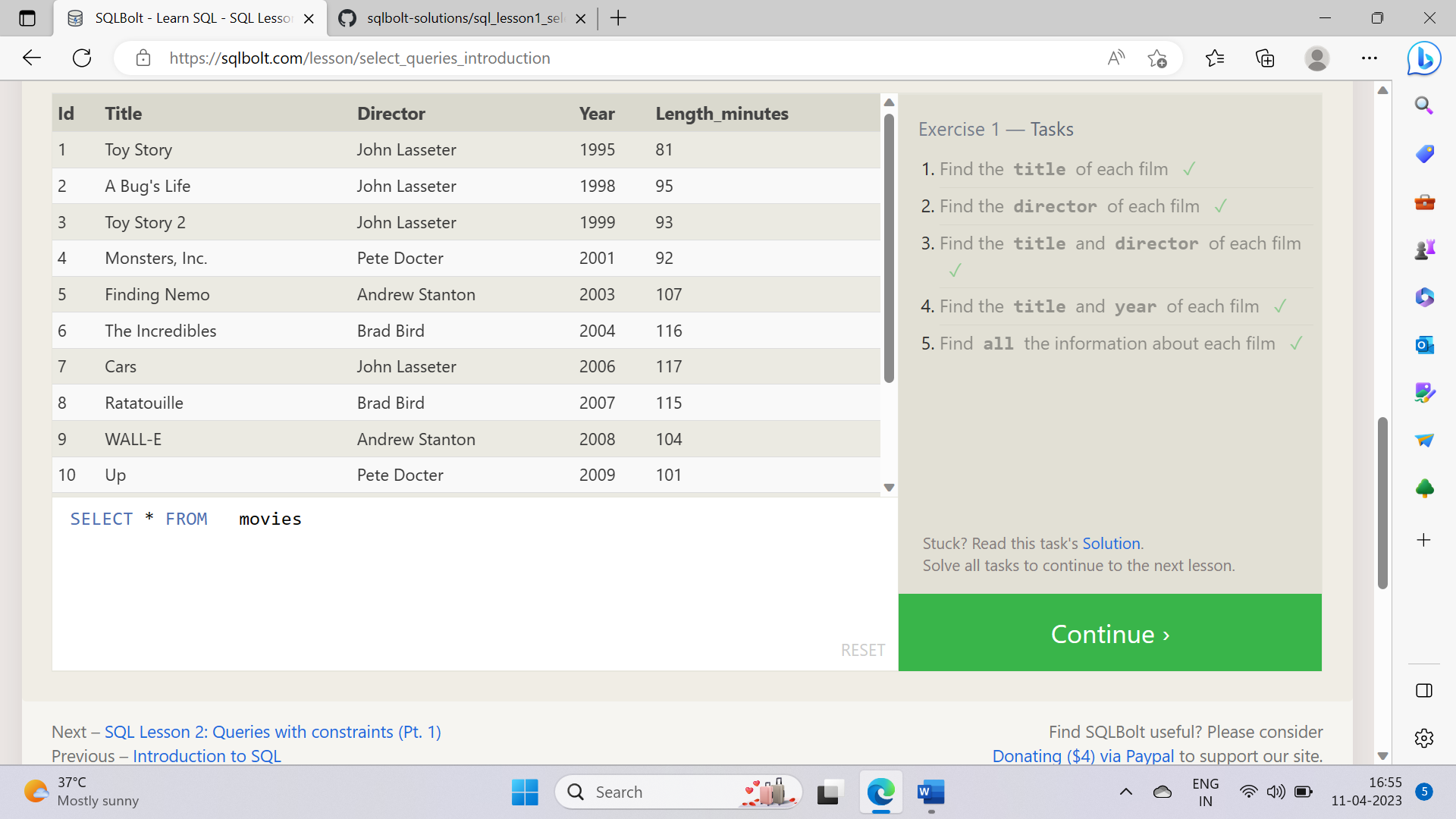
SQL Lesson 1: SELECT queries



1 . SELECT title FROM movies;

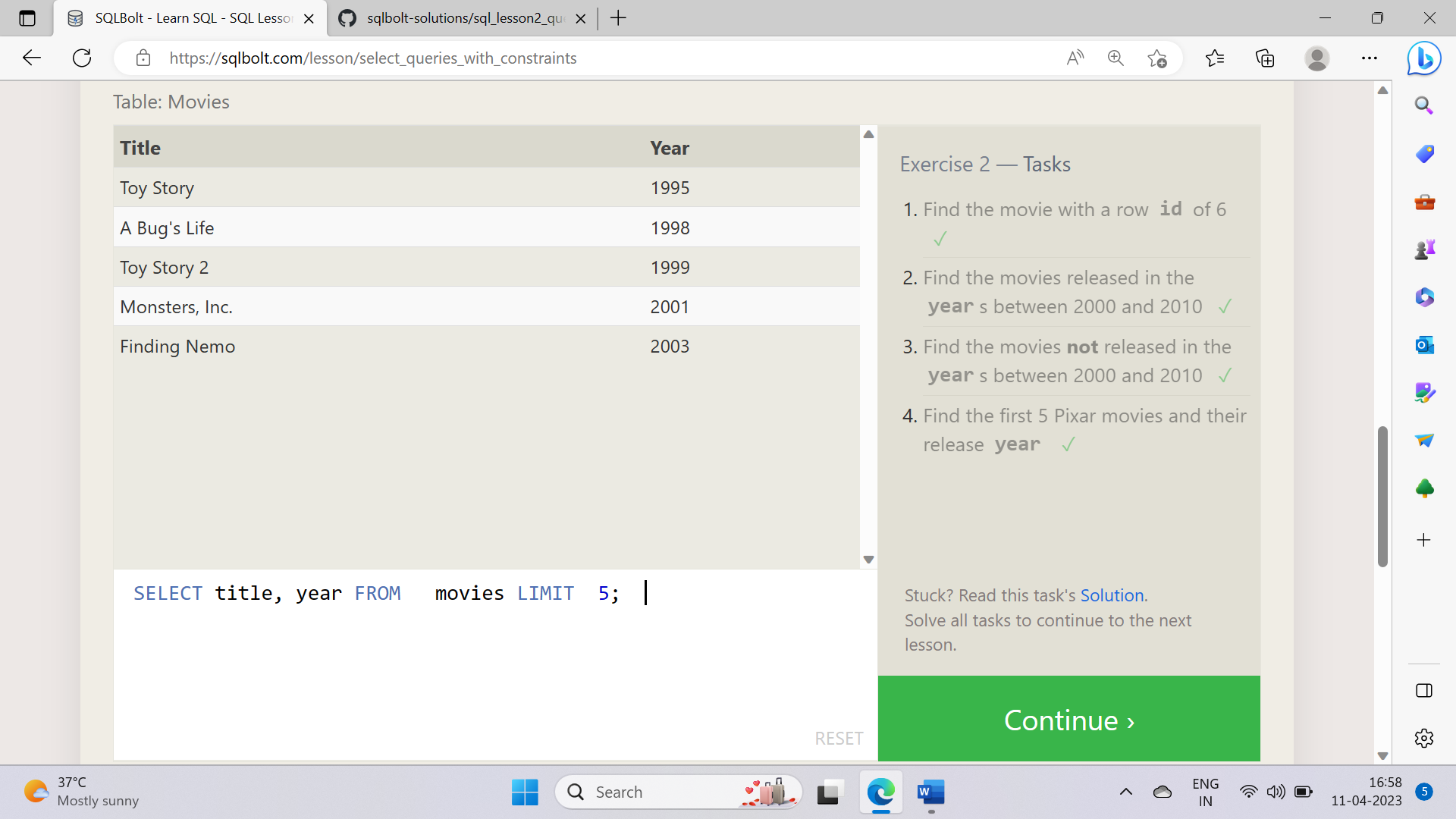
2 . SELECT director FROM movies;

3 . SELECT title, director FROM movies;

4 . SELECT title, year FROM movies;

5 . SELECT \* FROM movies

**SQL Lesson 2: Queries with constraints**



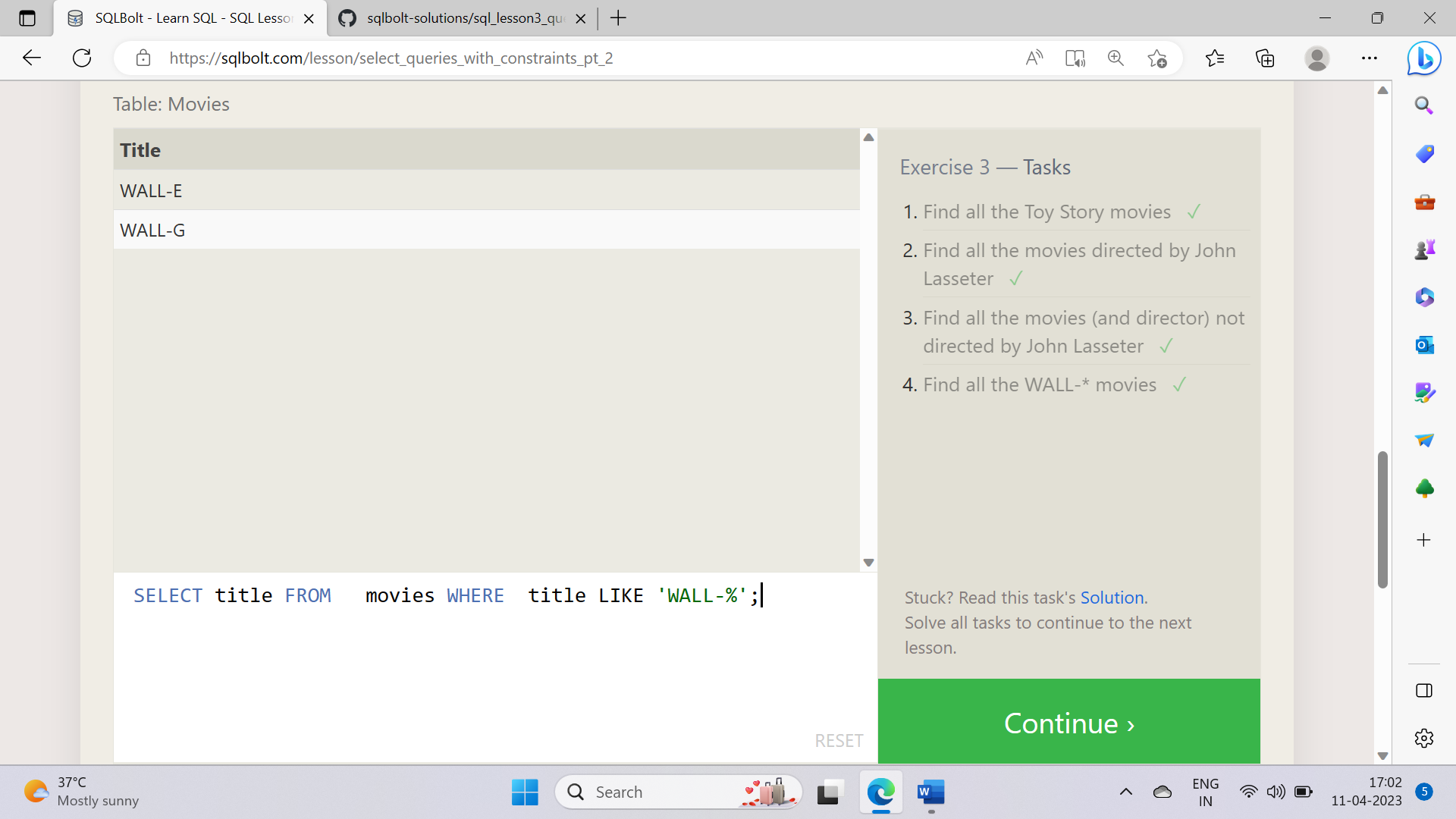
1 . SELECT \* FROM movies WHERE id = 6;

2 . SELECT \* FROM movies WHERE year BETWEEN 2000 AND 2010;

3 . SELECT \* FROM movies WHERE year NOT BETWEEN 2000 AND 2010;

4 . SELECT title, year FROM movies LIMIT 5;

**SQL Lesson 3: Queries with constraints**



1 . SELECT \* FROM movies WHERE title LIKE 'Toy Story%';

2 . SELECT \* FROM movies WHERE director = 'John Lasseter';

3 . SELECT title, director FROM movies WHERE director != 'John Lasseter';

4 . SELECT title FROM movies WHERE title LIKE 'WALL-%';

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



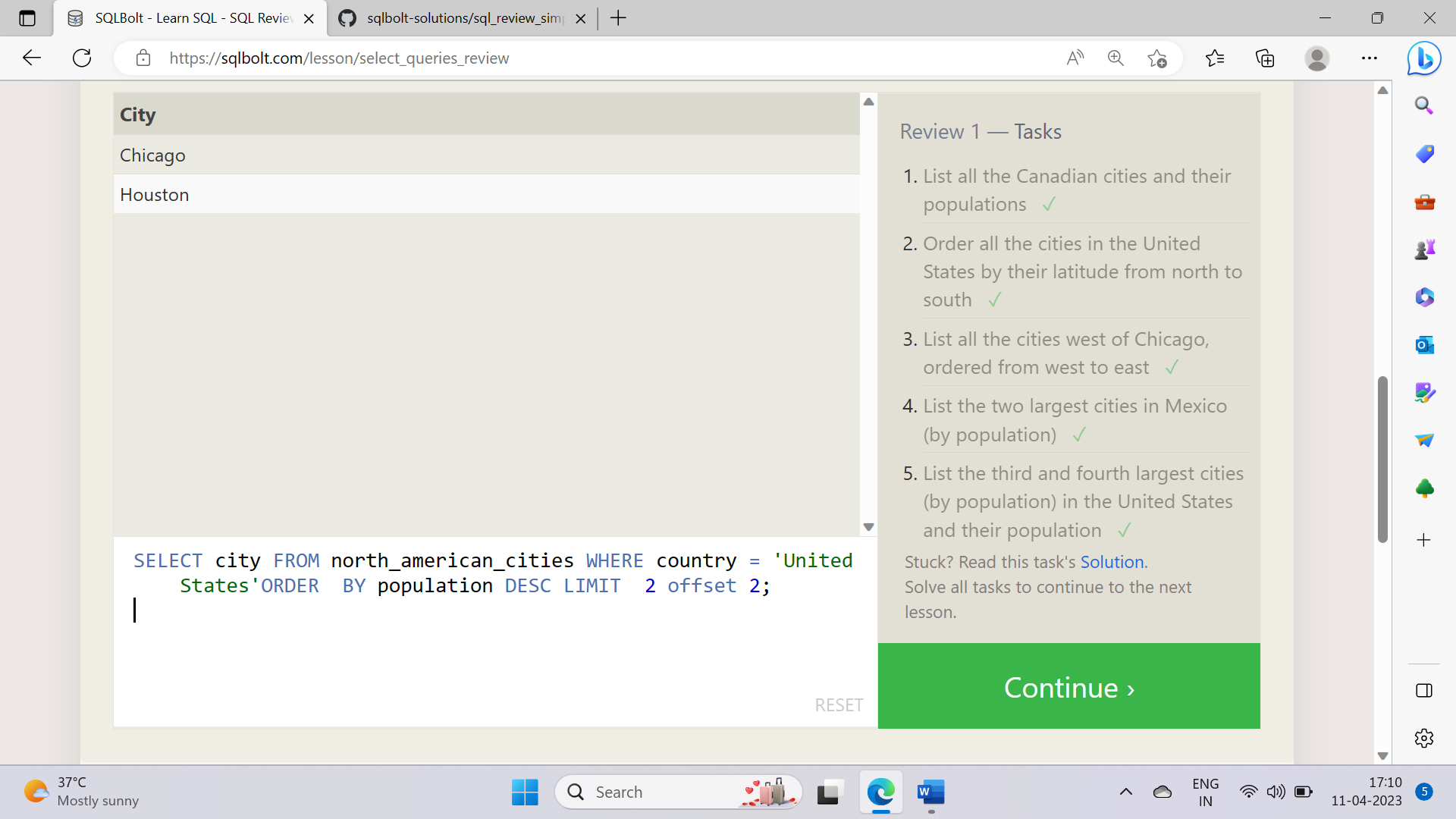
1 . SELECT DISTINCT director FROM movies ORDER BY director;

2 . SELECT title FROM movies ORDER BY year DESC LIMIT 4;

3 . SELECT title FROM movies ORDER BY title LIMIT 5;

4 . SELECT title FROM movies ORDER BY title LIMIT 5 offset 5;

**SQL Review: Simple SELECT Queries**



1 . SELECT city, population FROM north\_american\_cities WHERE country = 'Canada';

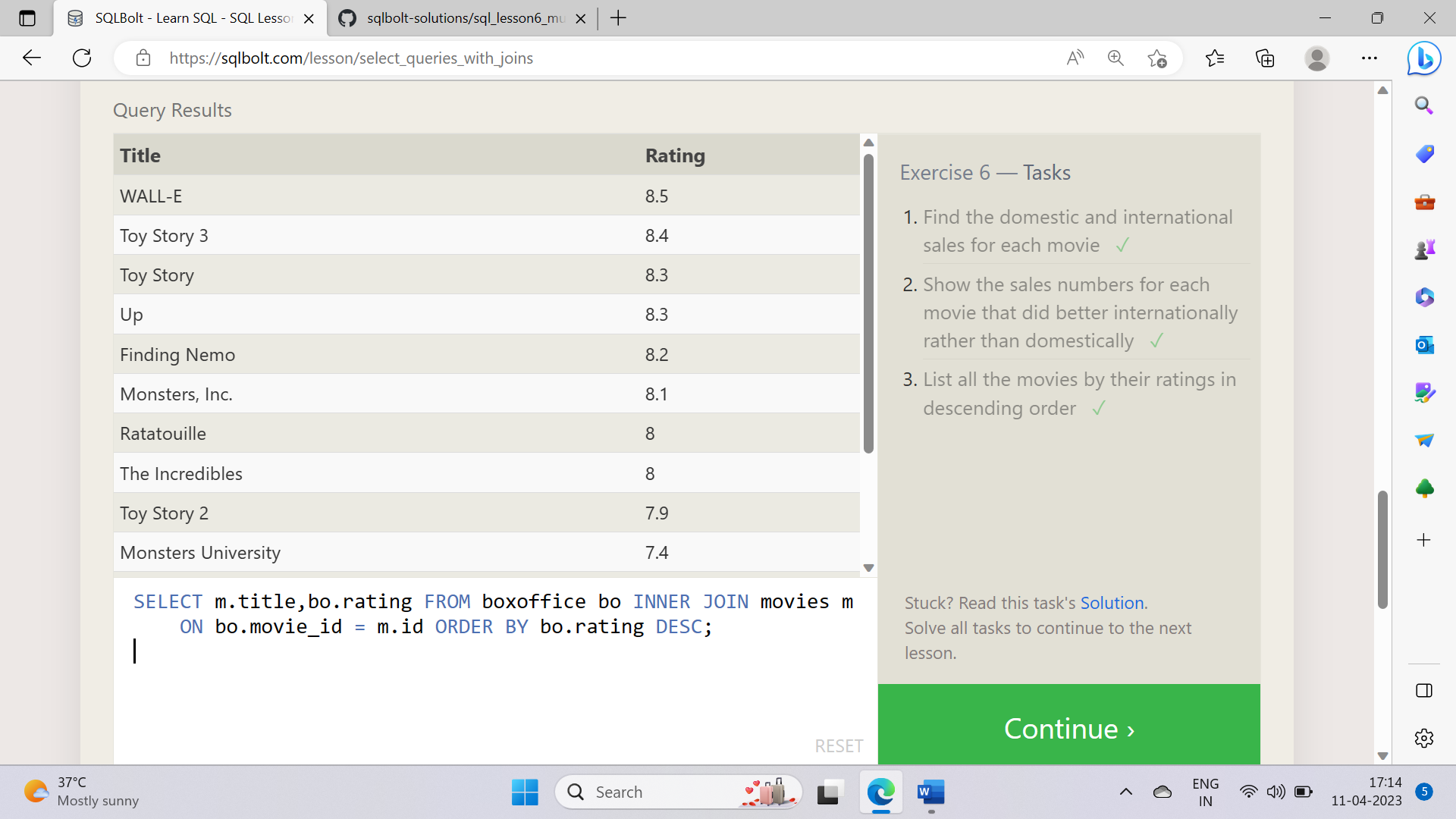
2 . SELECT city FROM north\_american\_cities WHERE country = 'United States' ORDER BY latitude DESC;

3 . SELECT city FROM north\_american\_cities ORDER BY longitude LIMIT 6;

4 . SELECT city FROM north\_american\_cities WHERE country = 'Mexico'ORDER BY population DESC LIMIT 2;

5 . SELECT city FROM north\_american\_cities WHERE country = 'United States'ORDER BY population DESC LIMIT 2 offset 2;

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

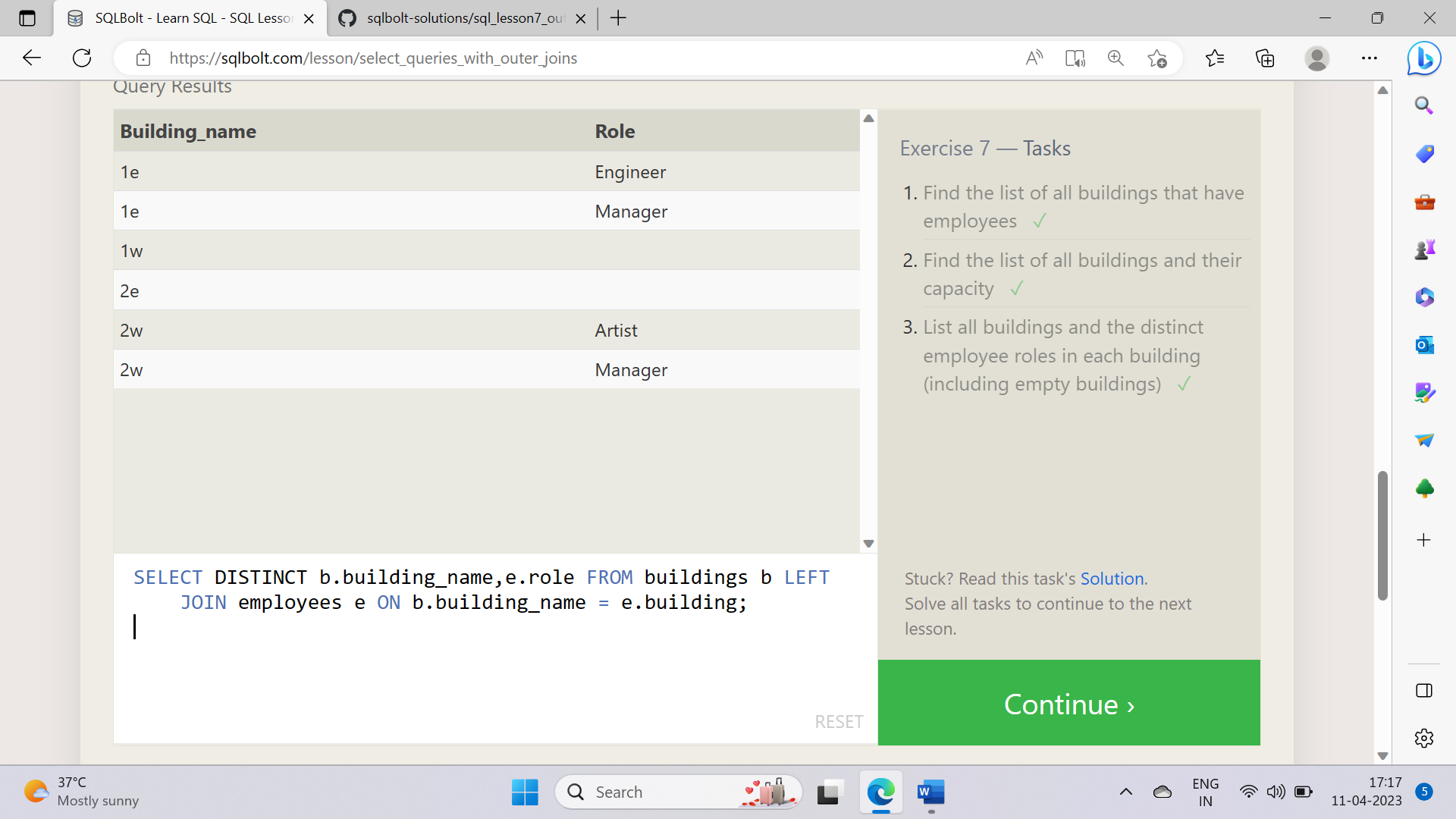


1 . SELECT m.title,bo.domestic\_sales,bo.international\_sales FROM boxoffice bo INNER JOIN movies m ON bo.movie\_id = m.id;

2 . SELECT m.title,bo.domestic\_sales,bo.international\_sales FROM boxoffice bo INNER JOIN movies m ON bo.movie\_id = m.id WHERE bo.international\_sales > bo.domestic\_sales;

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

**SQL Lesson 7: OUTER JOINs**

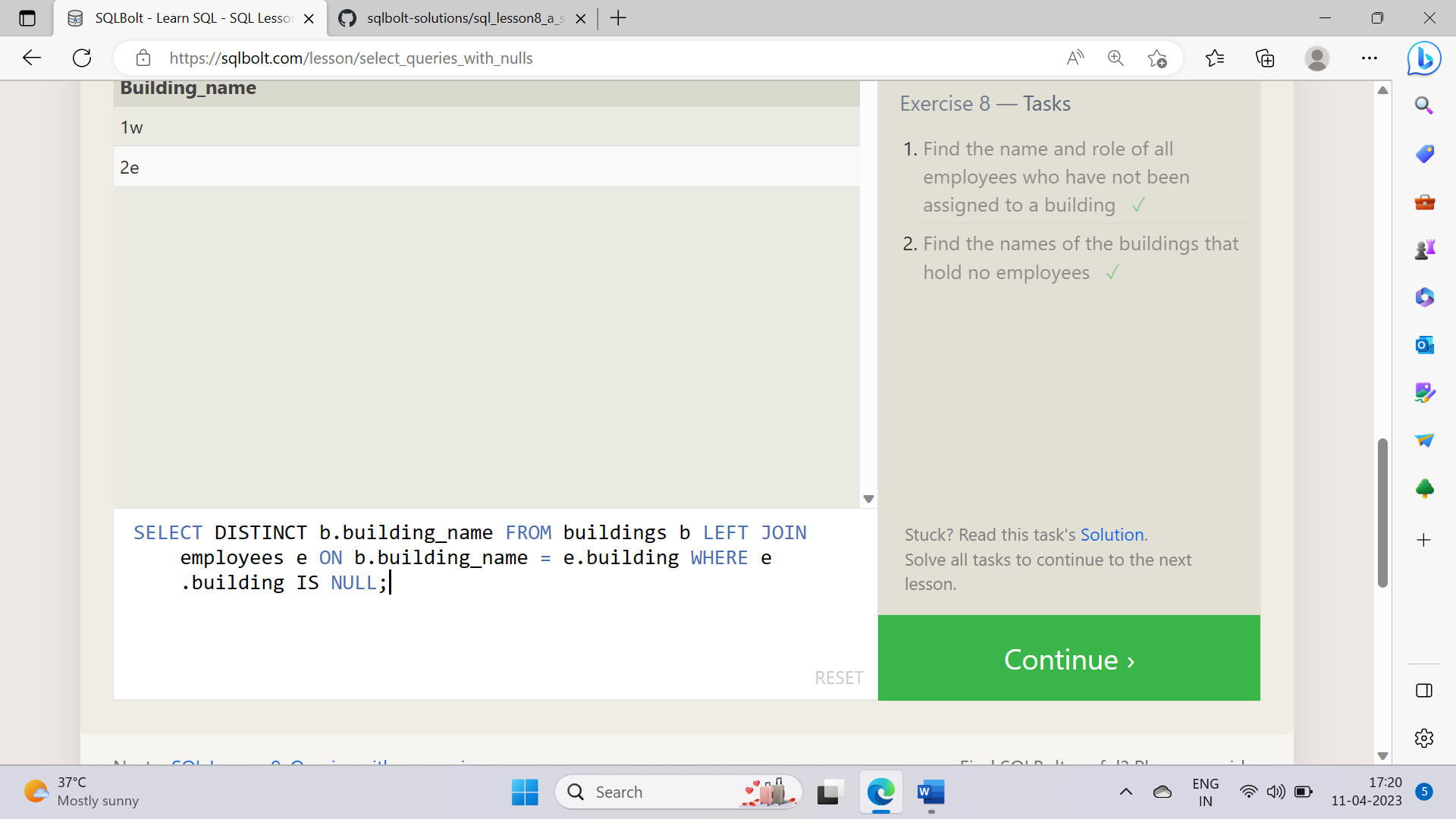


1 . SELECT DISTINCT b.building\_name FROM buildings b INNER JOIN employees e ON b.building\_name = e.building;

2 . SELECT \* FROM buildings;

3 . SELECT DISTINCT b.building\_name,e.role FROM buildings b LEFT JOIN employees e ON b.building\_name = e.building;

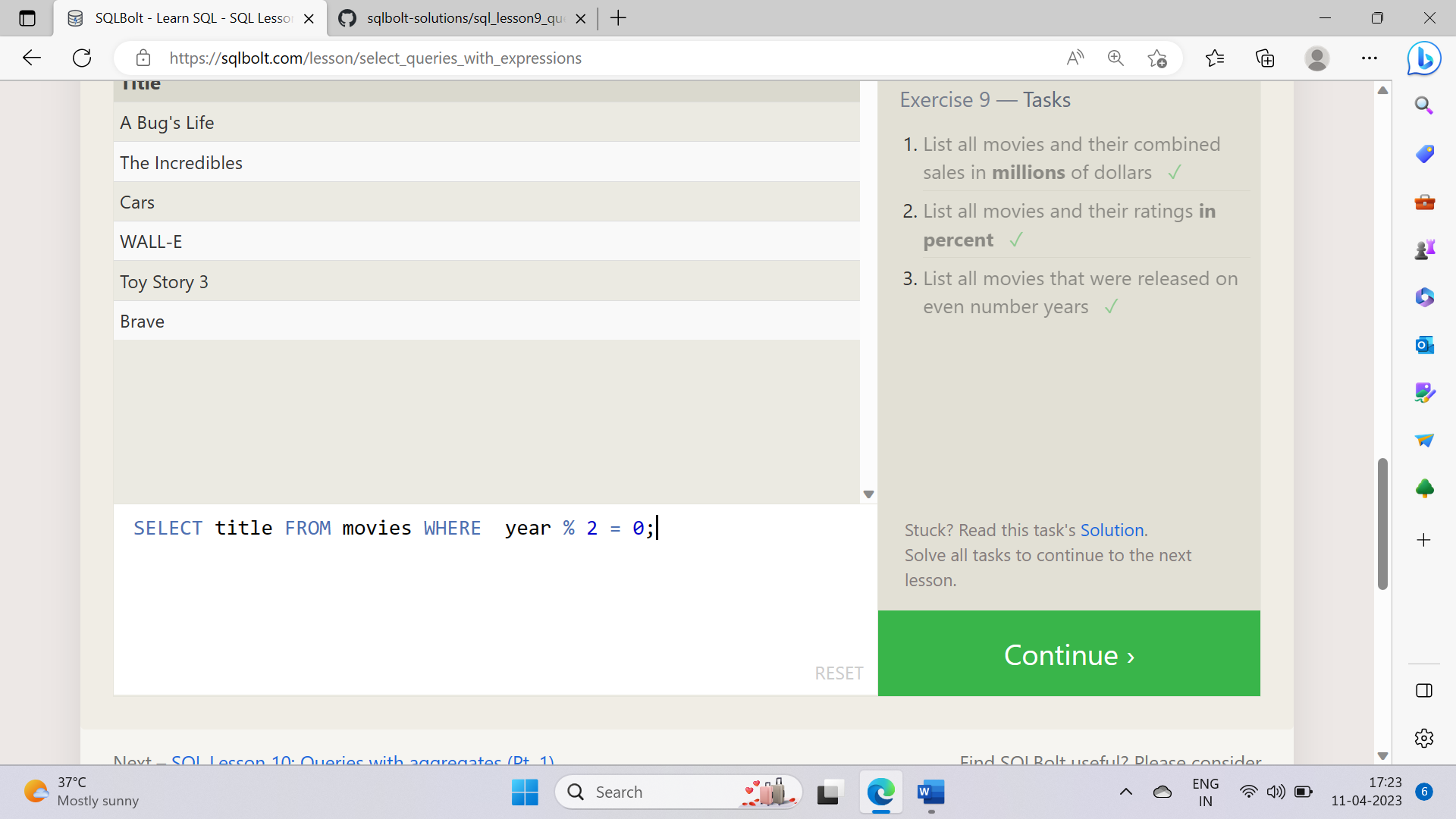
**SQL Lesson 8: A short note on NULLs**



1 . SELECT name,role FROM employees WHERE building IS NULL;

2 . SELECT DISTINCT b.building\_name FROM buildings b LEFT JOIN employees e ON b.building\_name = e.building WHERE e.building IS NULL;

**SQL Lesson 9: Queries with expressions**

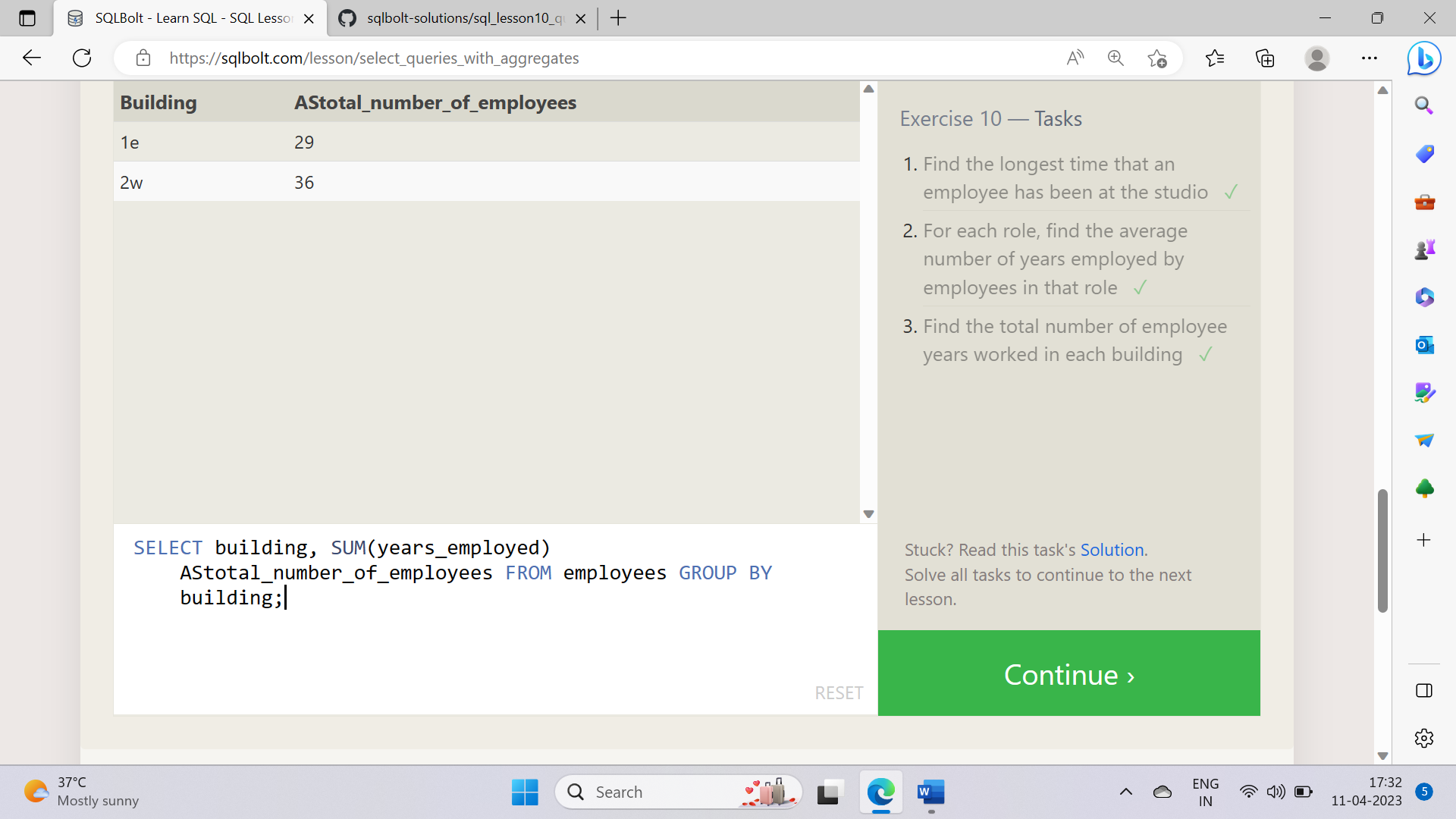


1 . SELECT m.title, ( bo.domestic\_sales + bo.international\_sales ) /1000000 AS combined\_sales\_in\_millions\_of\_dollars FROM movies m INNER JOIN boxoffice bo ON m.id = bo.movie\_id;

2 . SELECT m.title,( bo.rating \* 10 ) AS rating\_in\_perecent FROM movies m INNER JOIN boxoffice bo ON m.id = bo.movie\_id;

3 . SELECT title FROM movies WHERE year % 2 = 0;

**SQL Lesson 10: Queries with aggregates**

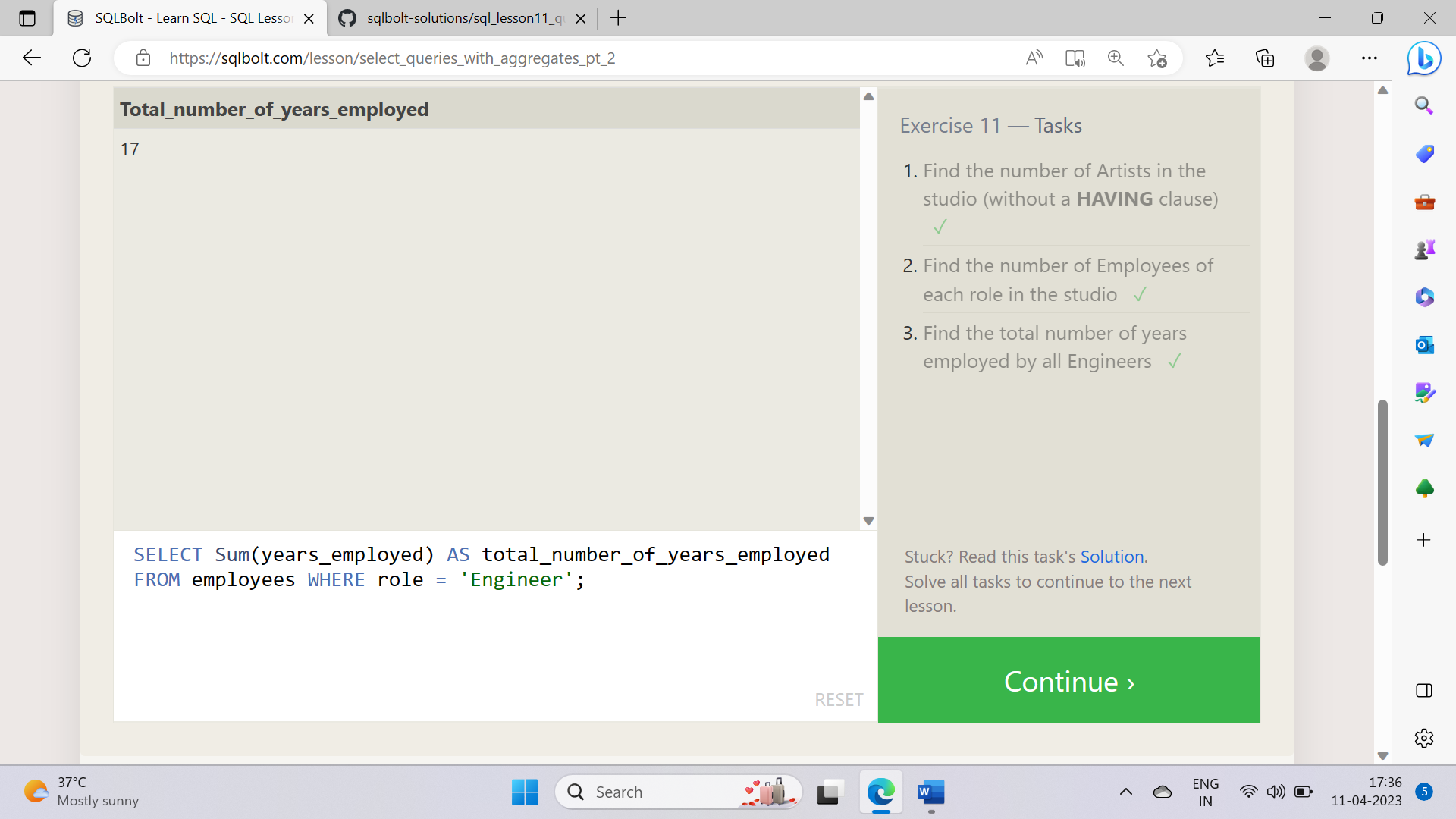


1 . SELECT Max (years\_employed) FROM employees;

2 . SELECT role,Avg(years\_employed) AS average\_number\_of\_years\_employed FROM employees GROUP BY role;

3 . SELECT building, SUM(years\_employed) AStotal\_number\_of\_employees FROM employees GROUP BY building;

**SQL Lesson 11: Queries with aggregates**



1 . SELECT Count(\*) AS number\_of\_artists FROM employees WHERE role = 'Artist';

2 . SELECT role,Count(\*) AS number\_of\_employees FROM employees GROUP BY role;

3 . SELECT Sum(years\_employed) AS total\_number\_of\_years\_employed

FROM employees WHERE role = 'Engineer';