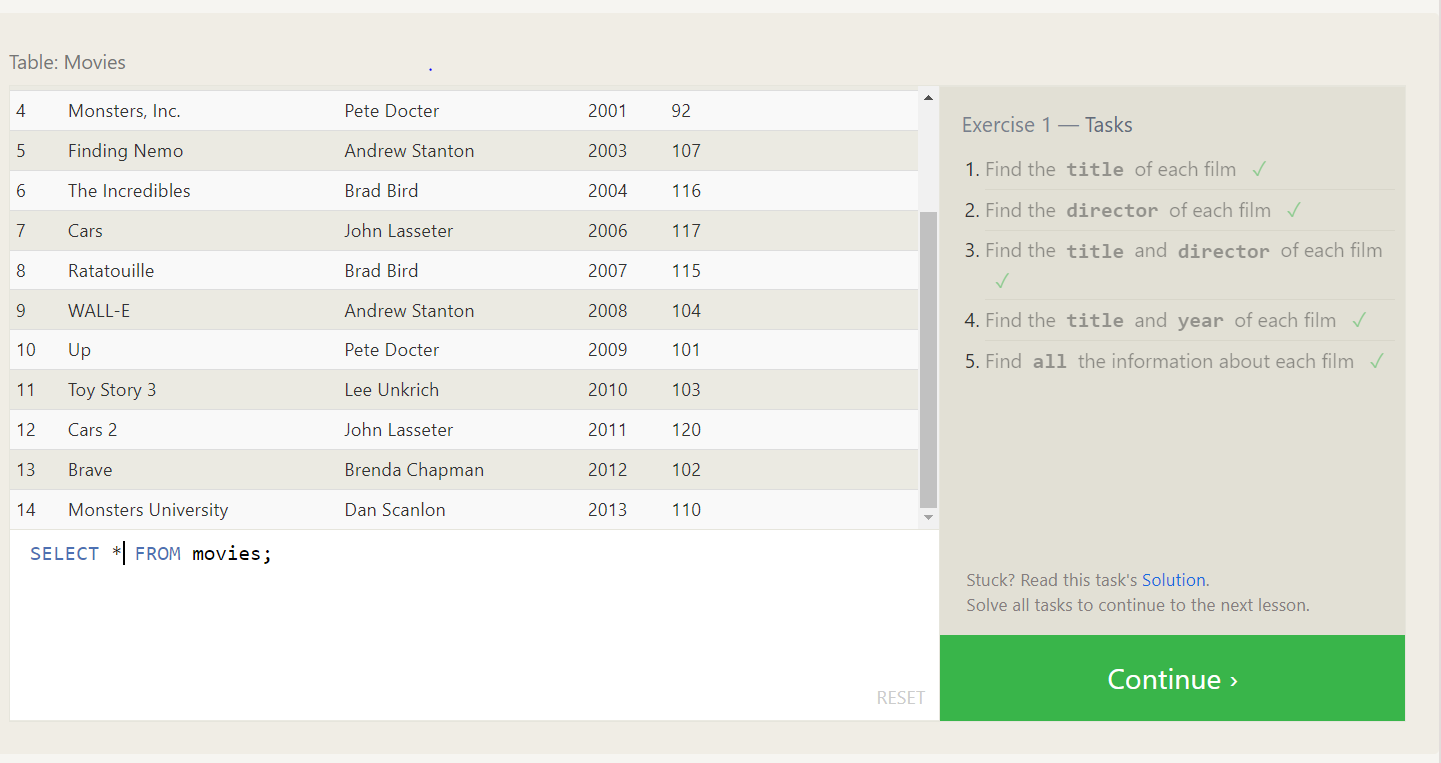
**SQL TASK 1**

**MANIKANDAN R**

**B53WDT**

**SQL Lesson 1: SELECT queries:**



1.SELECT title from movies;

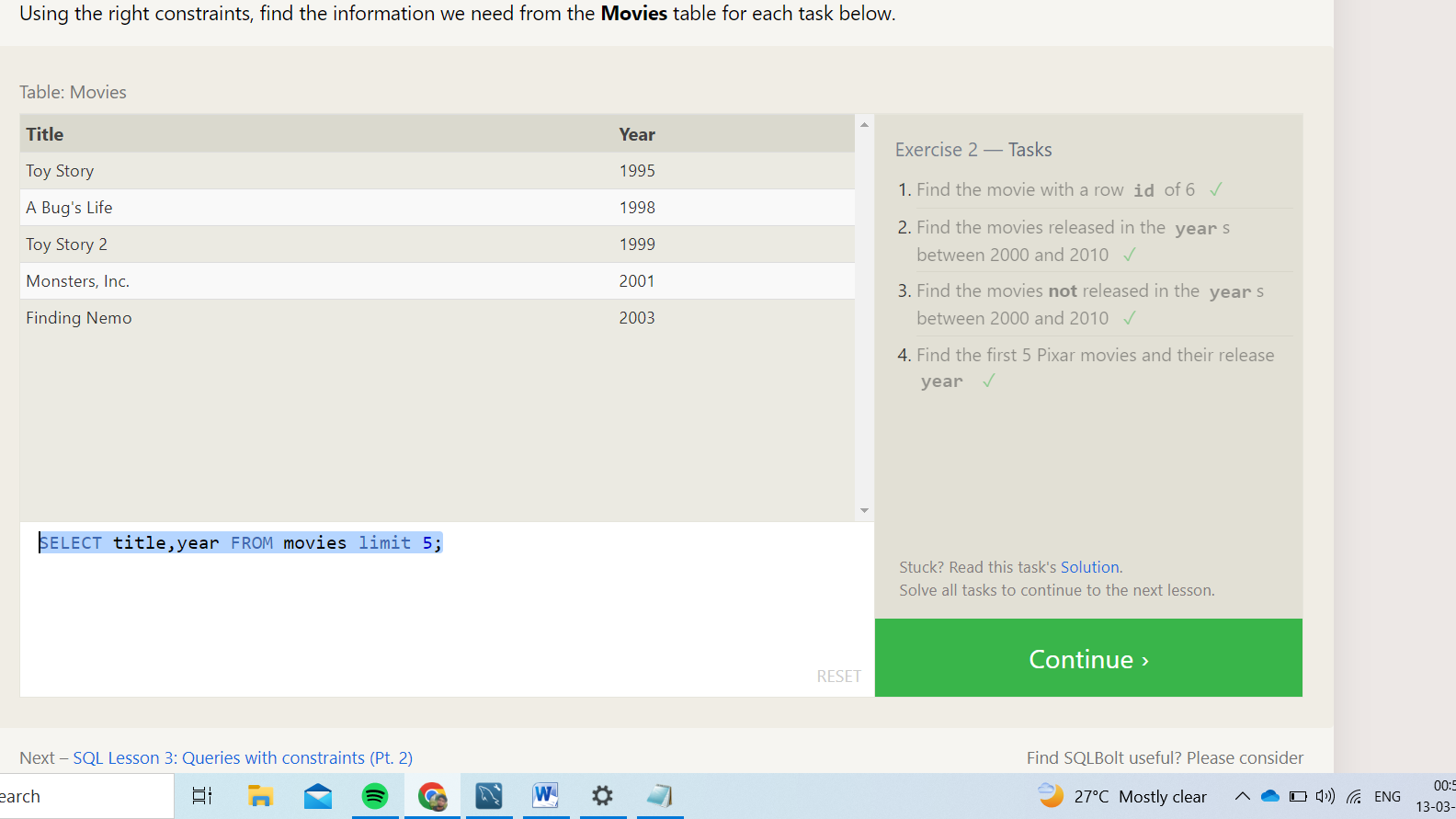
2.SELECT director from movies;

3.SELECT title,director from movies;

4.SELECT title,year from movies;

5.SELECT \* from movies;

**SQL Lesson 2: Queries with constraints:**



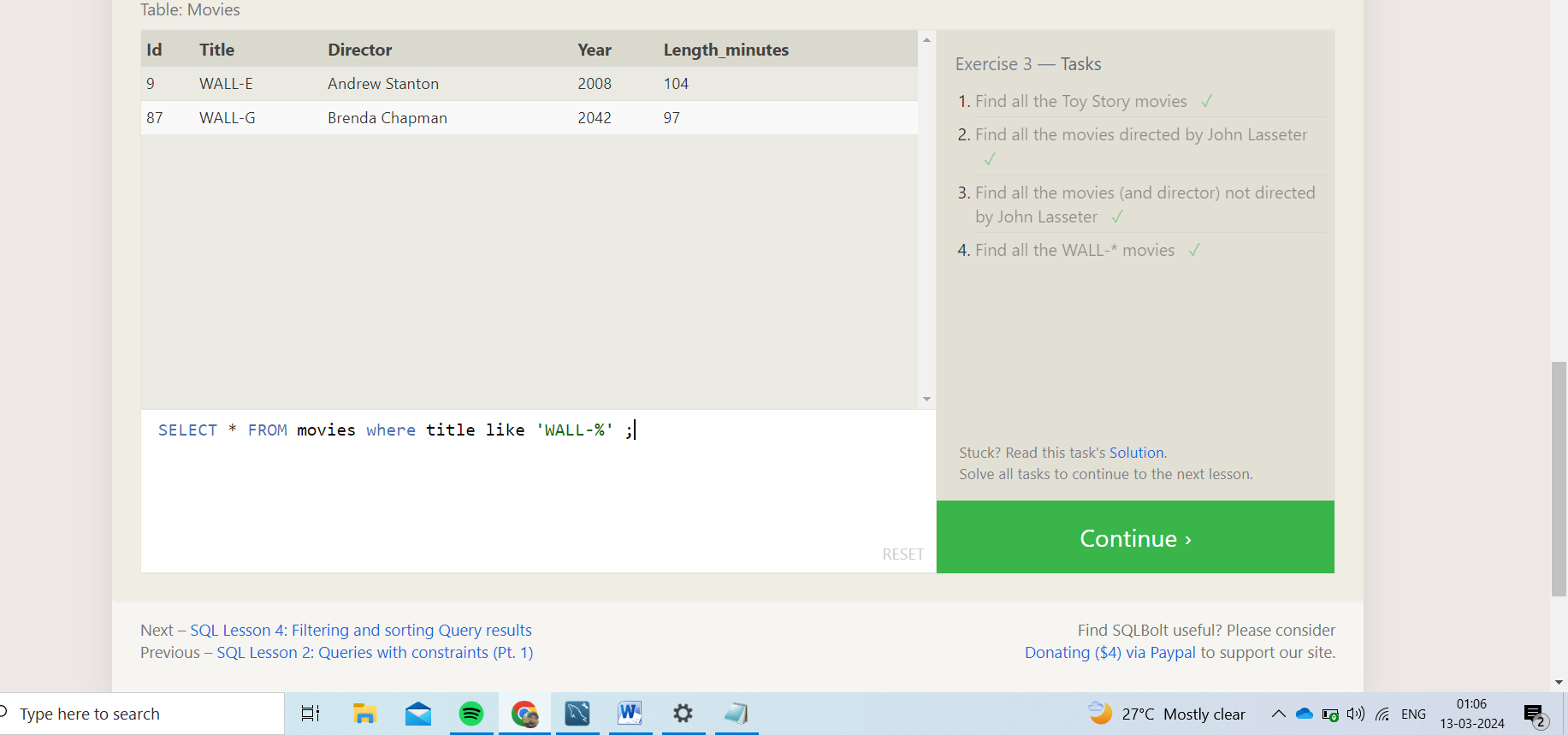
1.SELECT \* FROM movies where id = 6;

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**



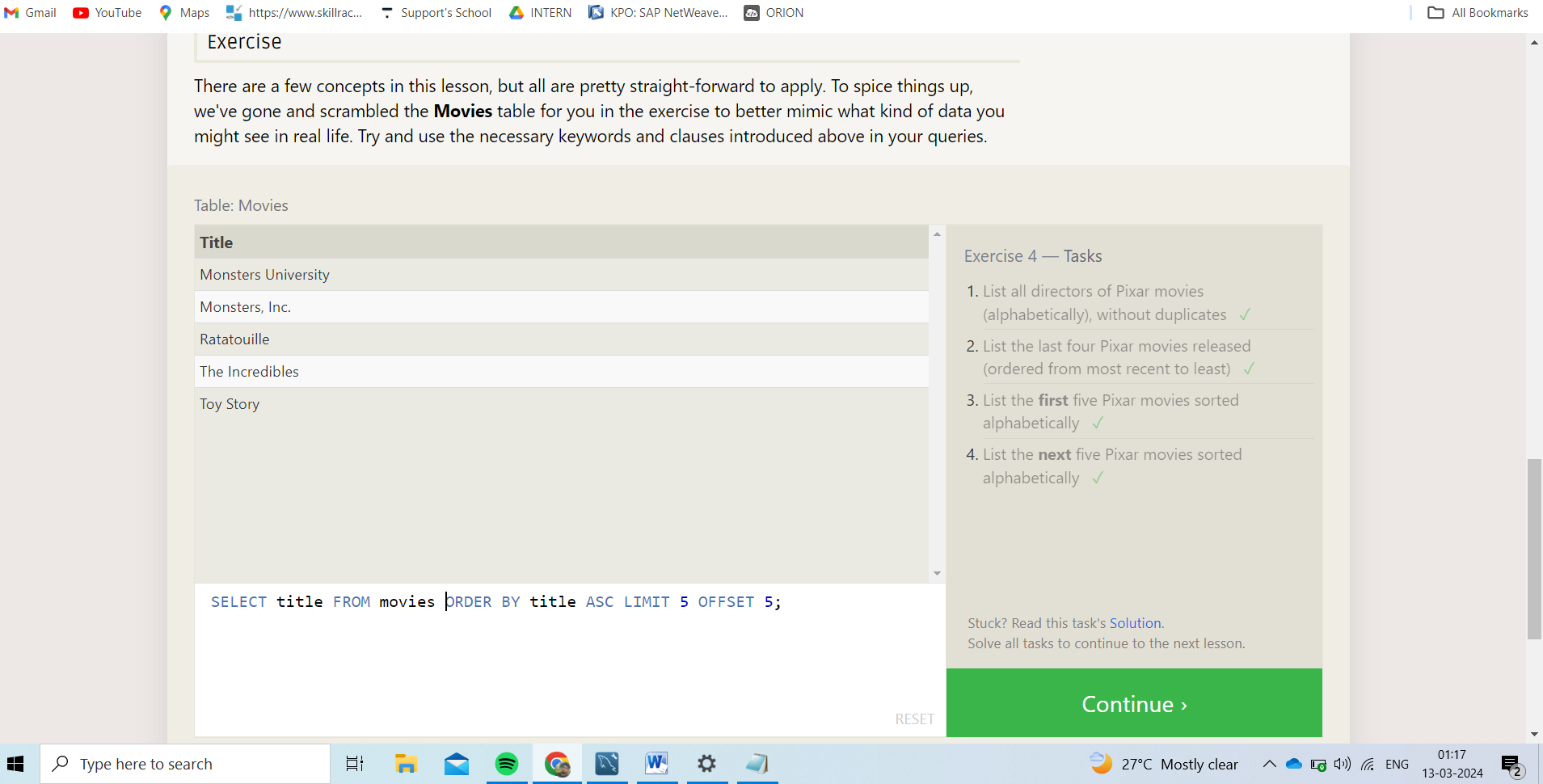
1.SELECT \* FROM movies where title like "Toy Story%";

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

3.SELECT \* FROM movies where director != 'John Lasseter';

4.SELECT \* FROM movies where title like 'WALL-%' ;

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



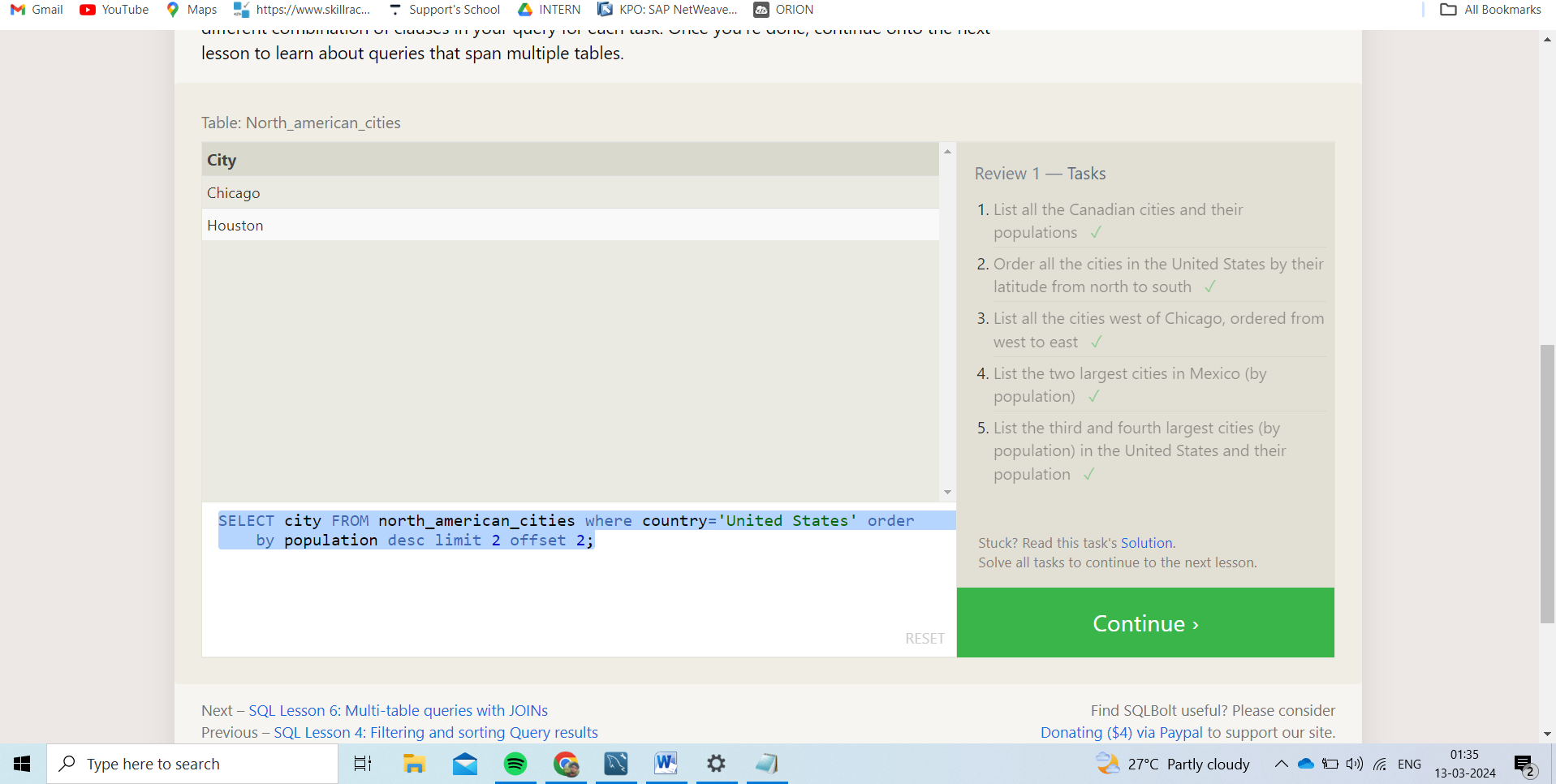
1.SELECT distinct(director) FROM movies order by director;

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 ASC LIMIT 5 OFFSET 5;

**SQL Review: Simple SELECT Queries:**



1.SELECT city,population FROM north\_american\_cities where country = "Canada";

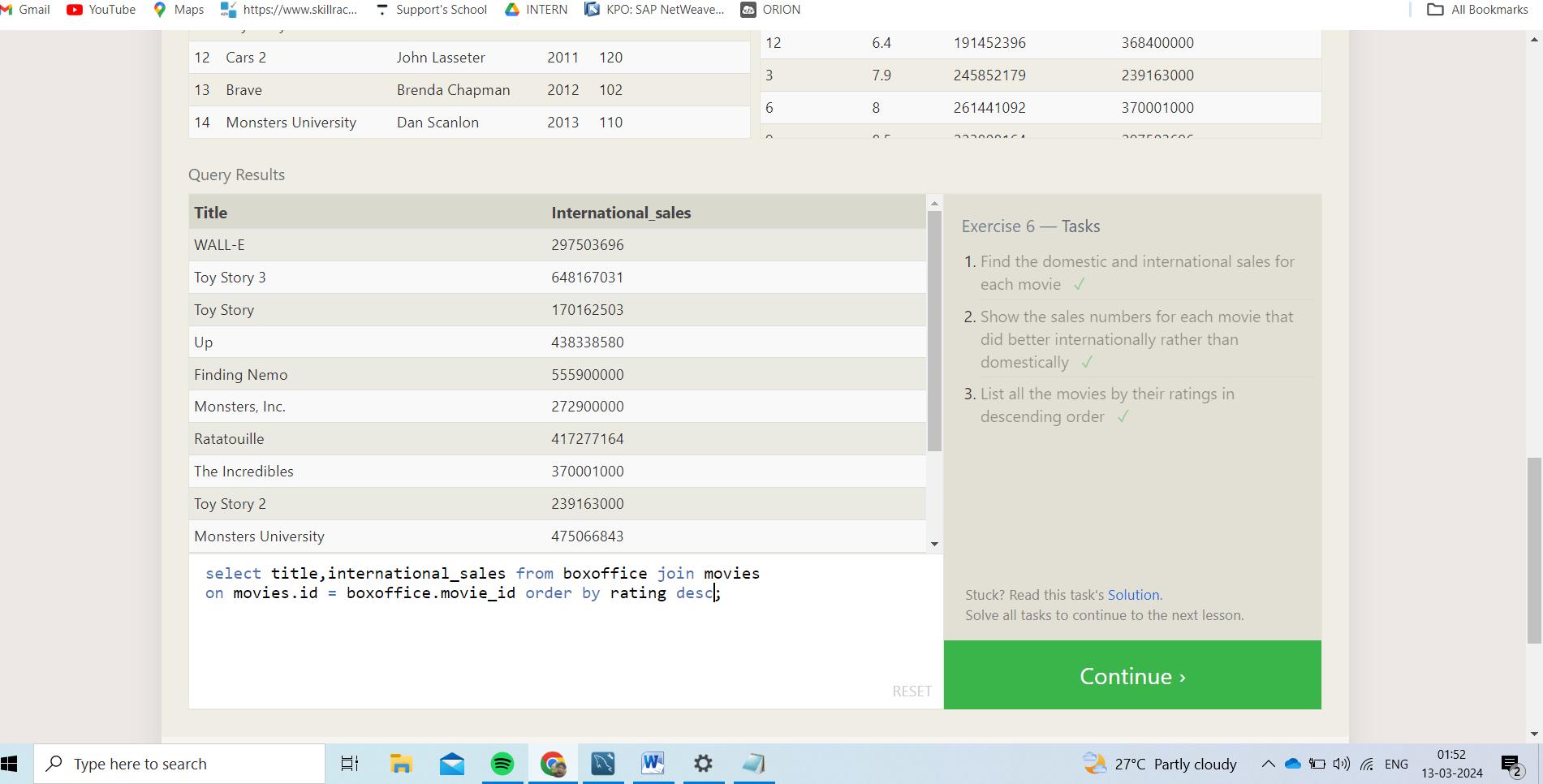
2.SELECT CITY FROM north\_american\_cities where country = "United States" ORDER BY LATITUDE DESC;

3.SELECT city, longitude FROM north\_american\_cities WHERE longitude < -87.629798 ORDER BY longitude ASC;

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**

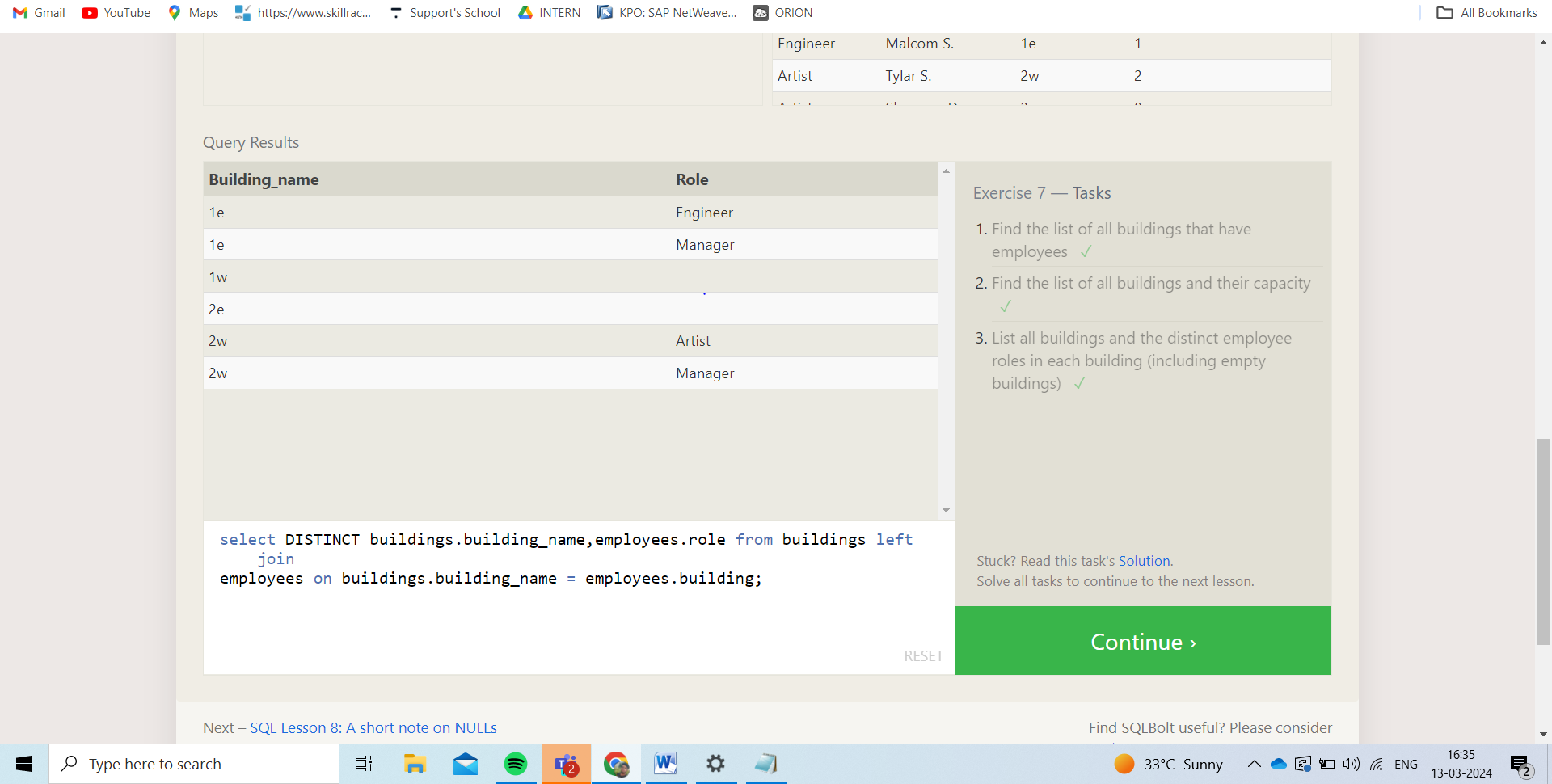


1.select title,domestic\_sales,international\_sales from boxoffice join movies on movies.id = boxoffice.movie\_id;

2.select title,domestic\_sales,international\_sales from boxoffice join movies on movies.id = boxoffice.movie\_id where international\_sales > domestic\_sales;

3.select title,international\_sales from boxoffice join movies on movies.id = boxoffice.movie\_id order by rating desc;

**SQL Lesson 7: OUTER JOINs:**

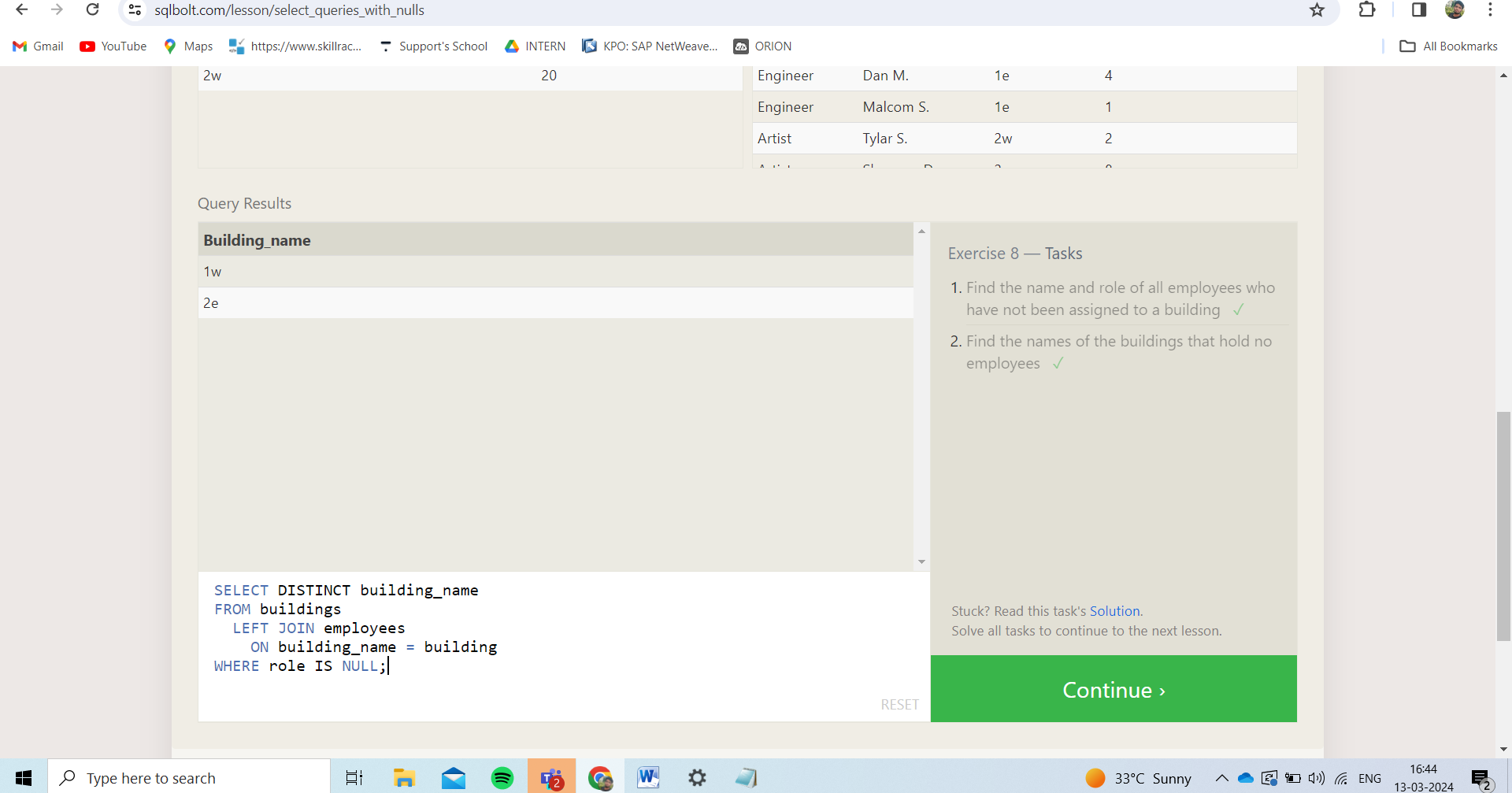


1.SELECT distinct(buildings.building\_name)FROM employees join Buildings on buildings.Building\_name = employees.Building;

2.SELECT building\_name,capacity FROM Buildings;

3.select DISTINCT buildings.building\_name,employees.role from buildings left join employees on buildings.building\_name = employees.building;

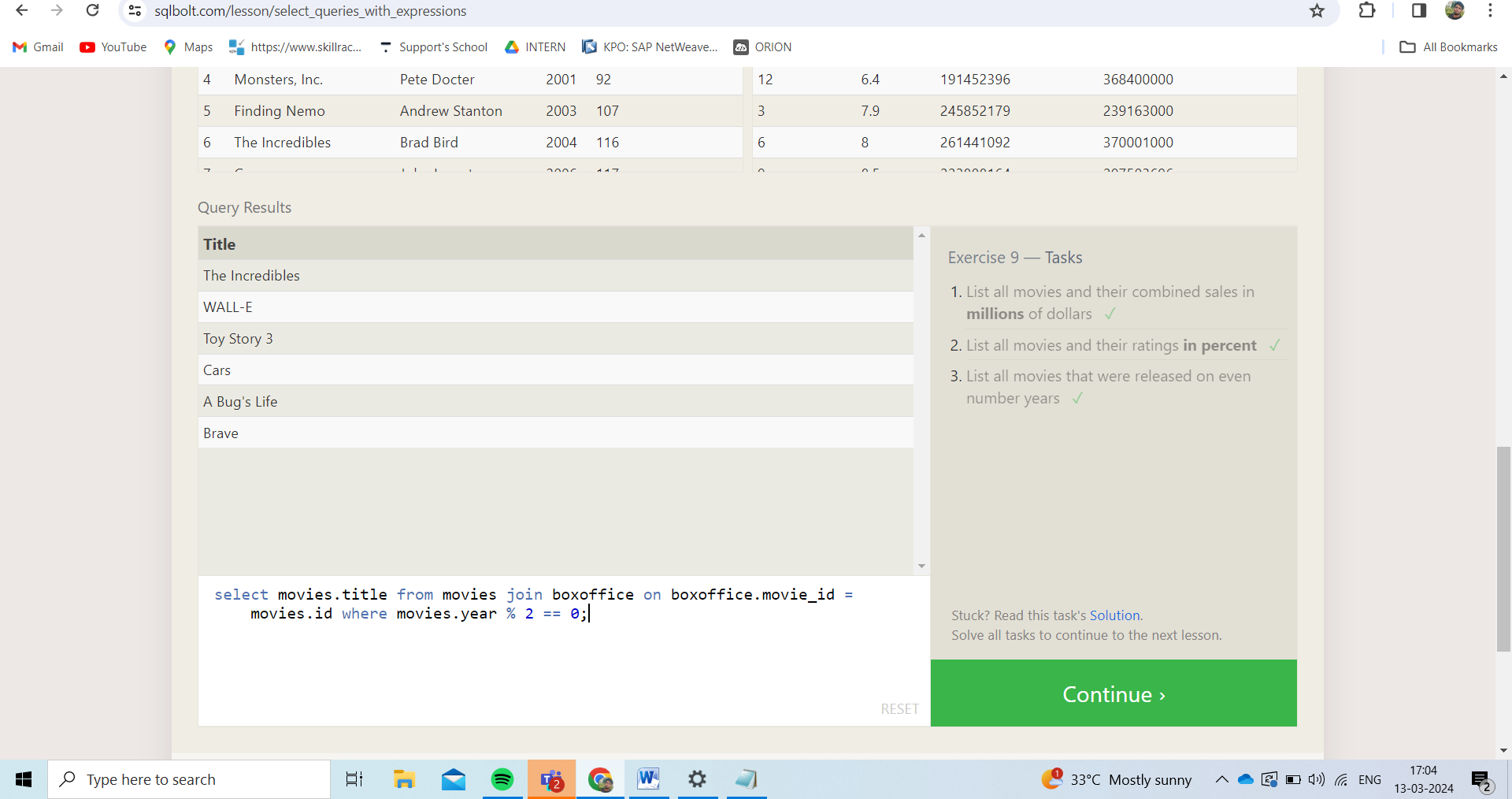
**SQL Lesson 8: A short note on NULLs:**



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;

**SQL Lesson 9: Queries with expressions:**

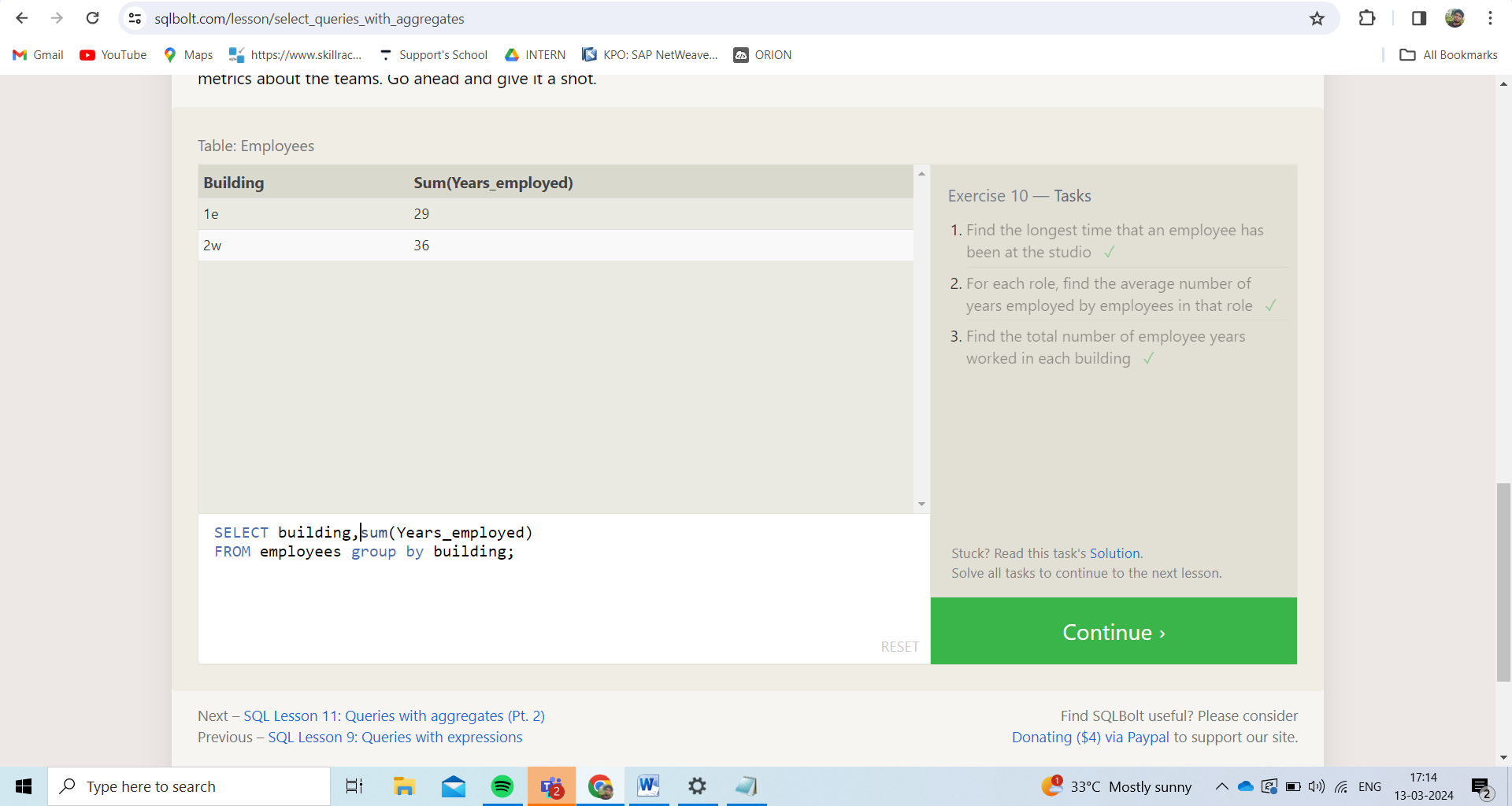


1.SELECT title,(boxoffice.international\_sales+boxoffice.domestic\_sales)/1000000 as combined\_sales FROM movies join boxoffice on movies.id = boxoffice.movie\_id;

2.select movies.title,boxoffice.rating\*10 as rating\_in\_percentage from movies join boxoffice on movies.id=boxoffice.Movie\_id;

3.select movies.title from movies join boxoffice on boxoffice.movie\_id = movies.id where movies.year % 2 == 0;

**SQL Lesson 10: Queries with aggregates:**

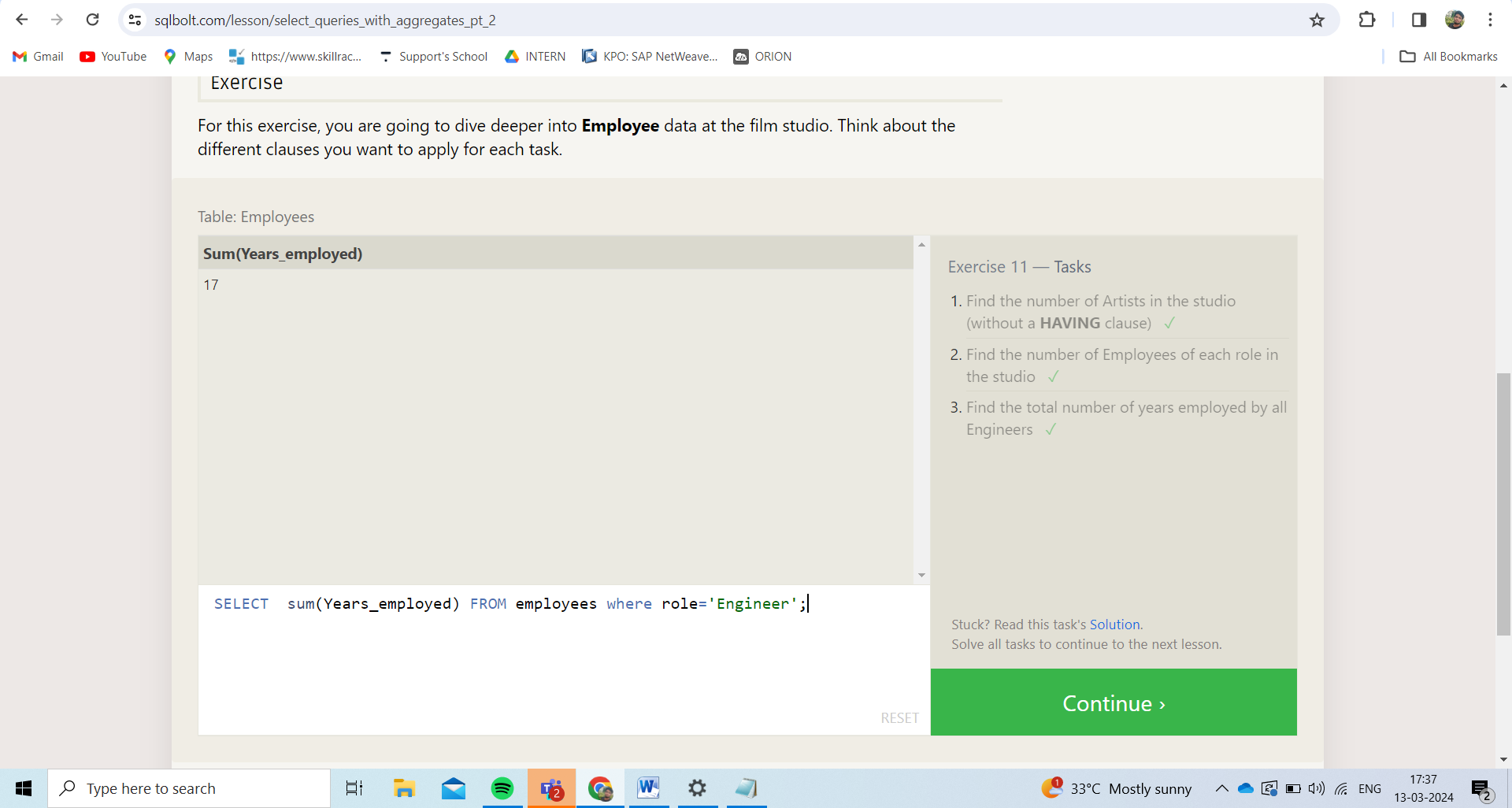


1.SELECT max(Years\_employed),name FROM employees;

2.SELECT role, AVG(years\_employed) as Average\_years\_employed FROM employees GROUP BY role;

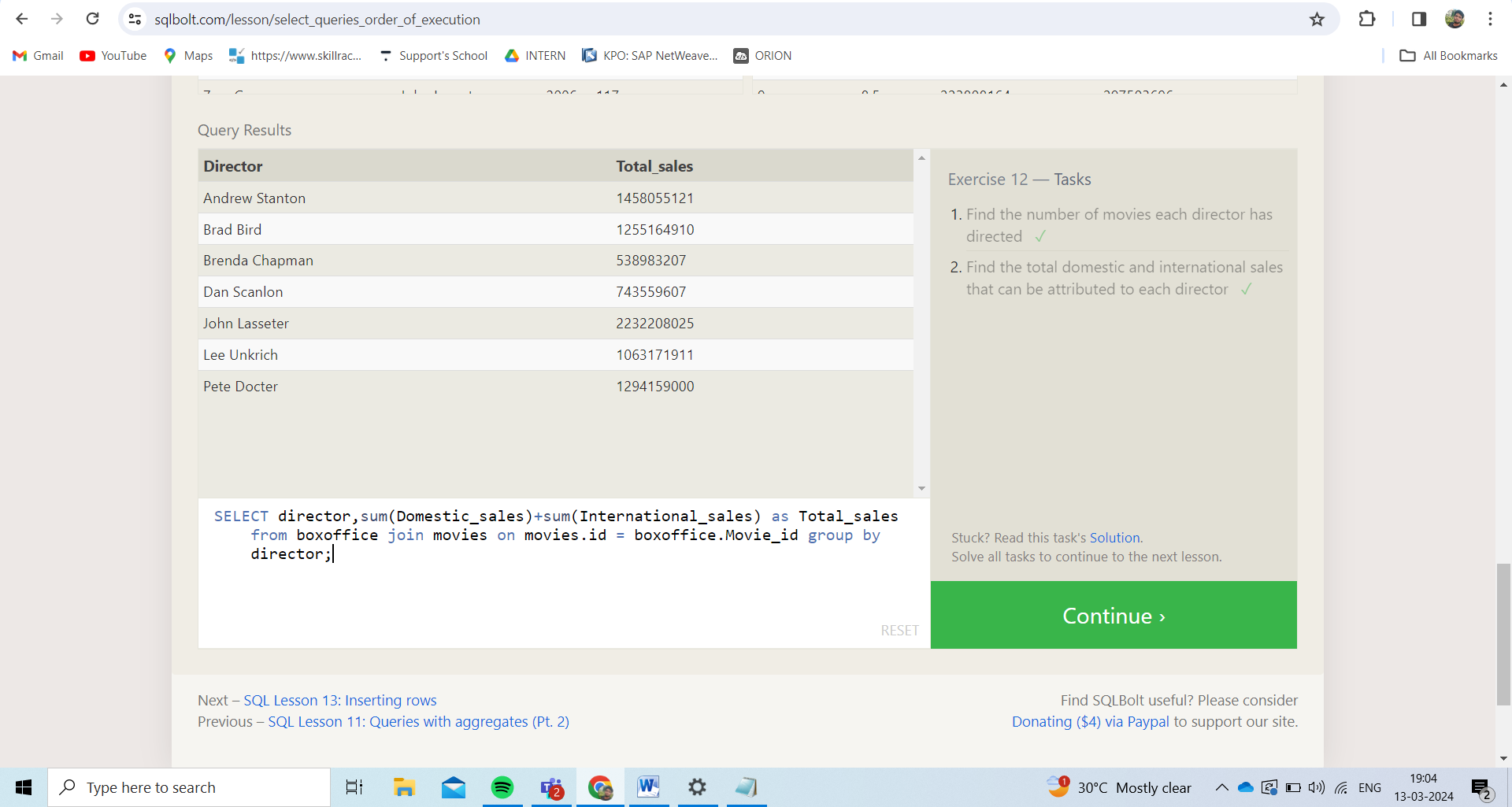
3.SELECT building,sum(Years\_employed) FROM employees group by building;

**SQL Lesson 11: Queries with aggregates:**



1. SELECT count(role) FROM employees where role = 'Artist';
2. SELECT role,count(name) FROM employees group by role;
3. SELECT sum(Years\_employed) FROM employees where role='Engineer';

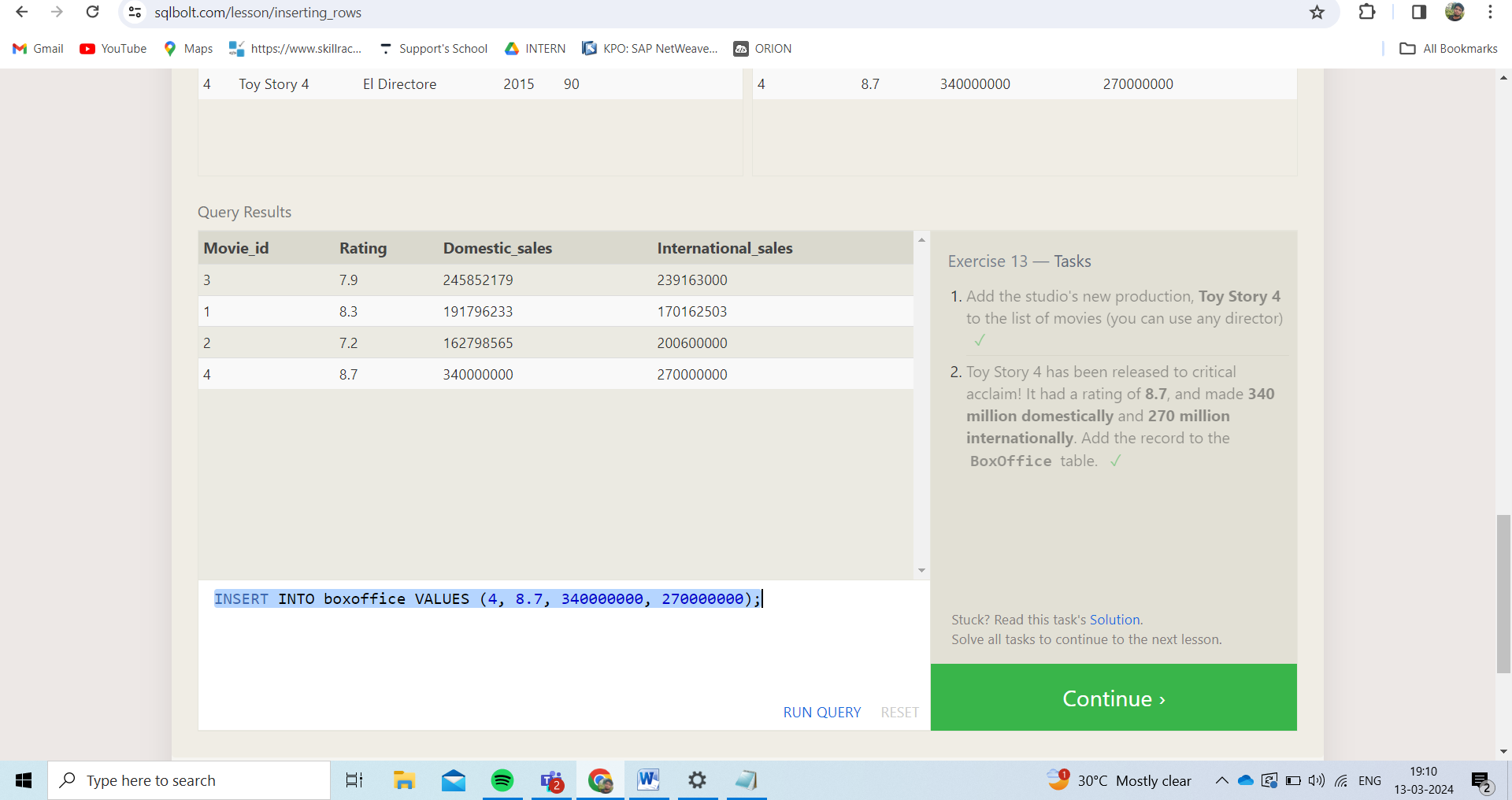
**SQL Lesson 12: Order of execution of a Query:**



1.SELECT director,count(director) FROM movies group by director;

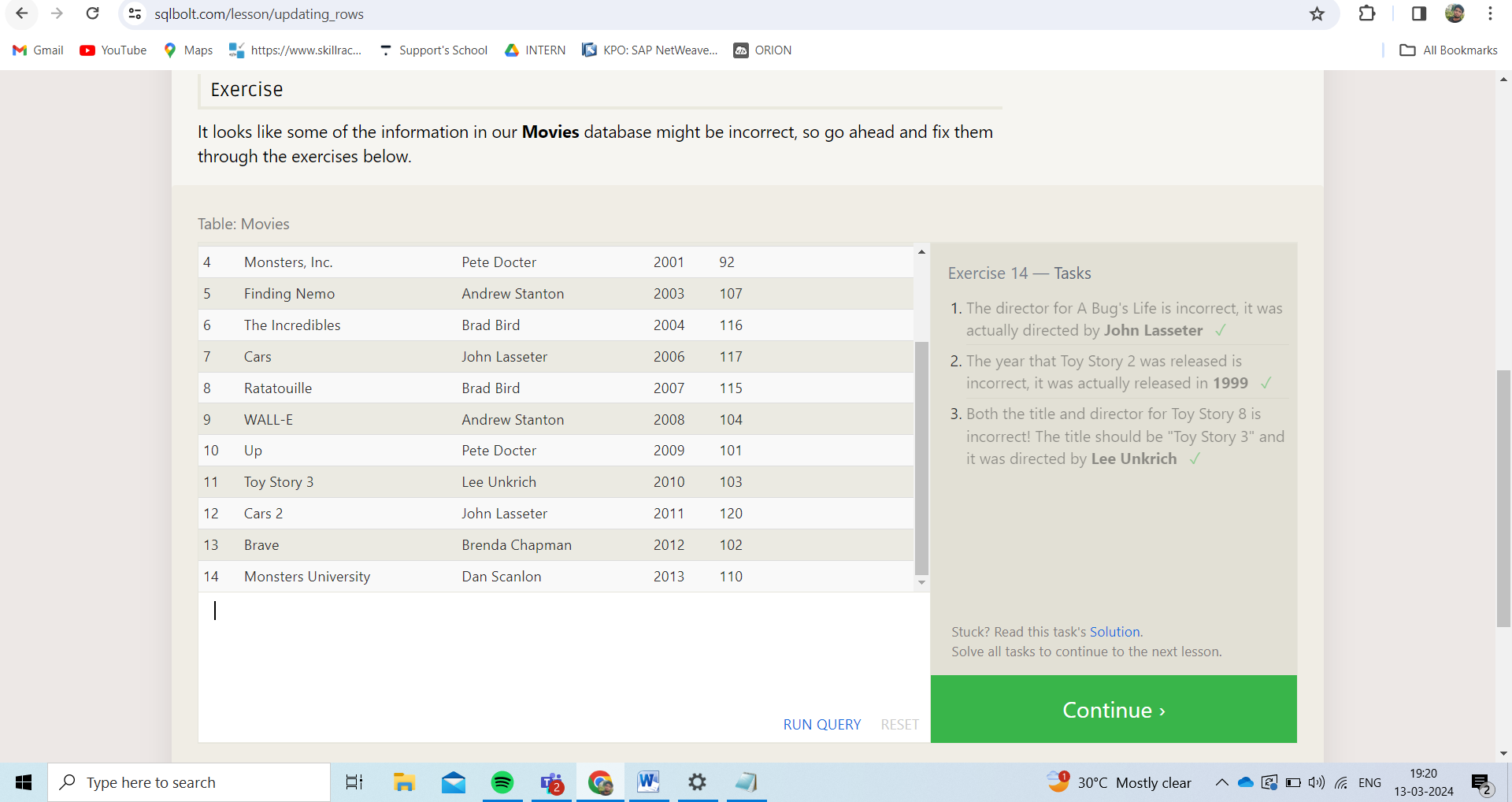
2. SELECT director,sum(Domestic\_sales)+sum(International\_sales) as Total\_sales from boxoffice join movies on movies.id = boxoffice.Movie\_id group by director;

**SQL Lesson 13: Inserting rows:**



1. INSERT INTO movies VALUES (4, "Toy Story 4", "El Directore", 2015, 90);
2. INSERT INTO boxoffice VALUES (4, 8.7, 340000000, 270000000);

**SQL Lesson 14: Updating rows:**

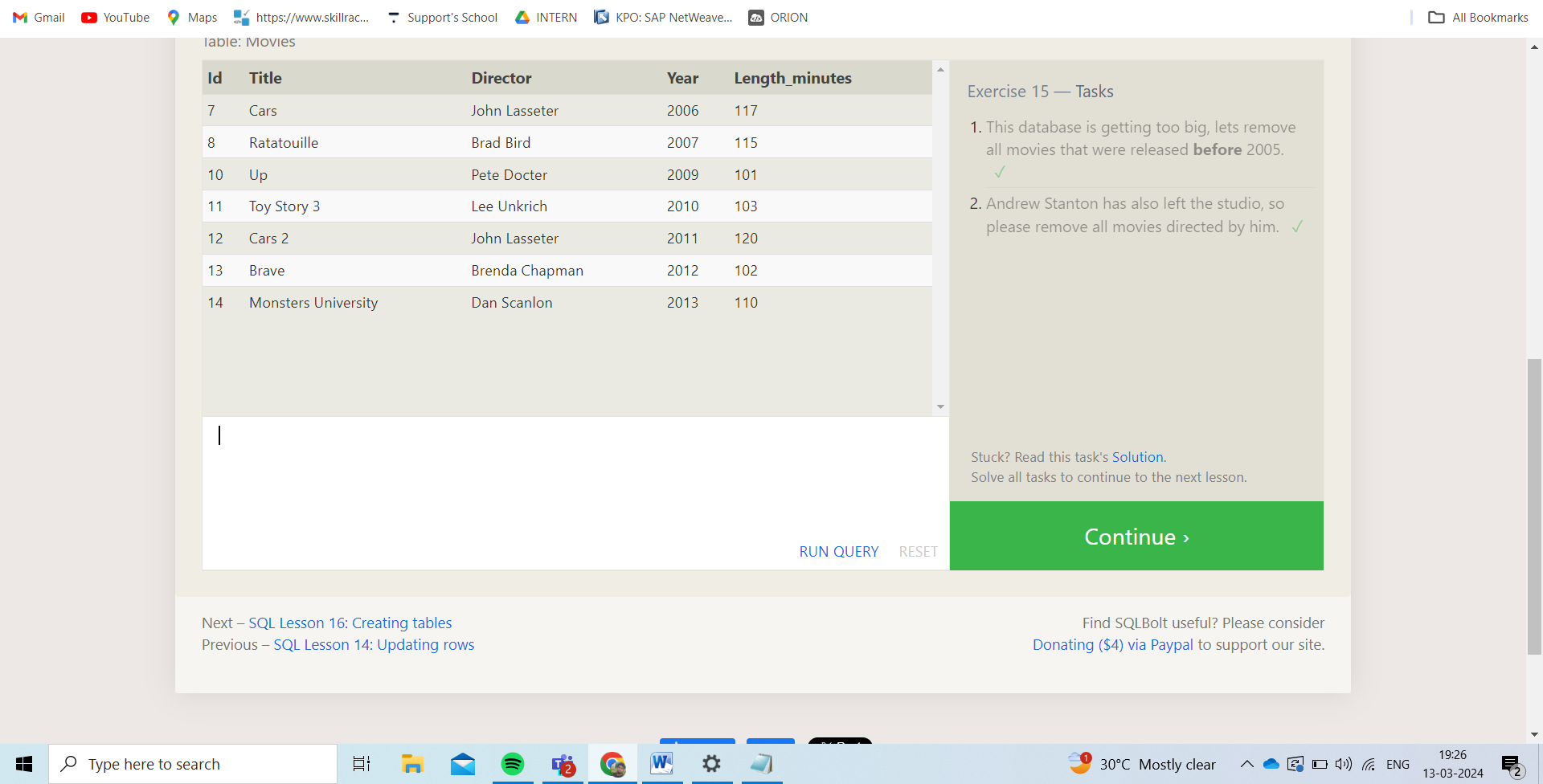


1.update movies set Director = 'John Lasseter' where title = "A Bug's Life";

2.update movies set year = '1990' where title = 'Toy Story 2';

3.update movies set title = 'Toy Story 3',Director = 'Lee Unkrich' where title = 'Toy Story 8';

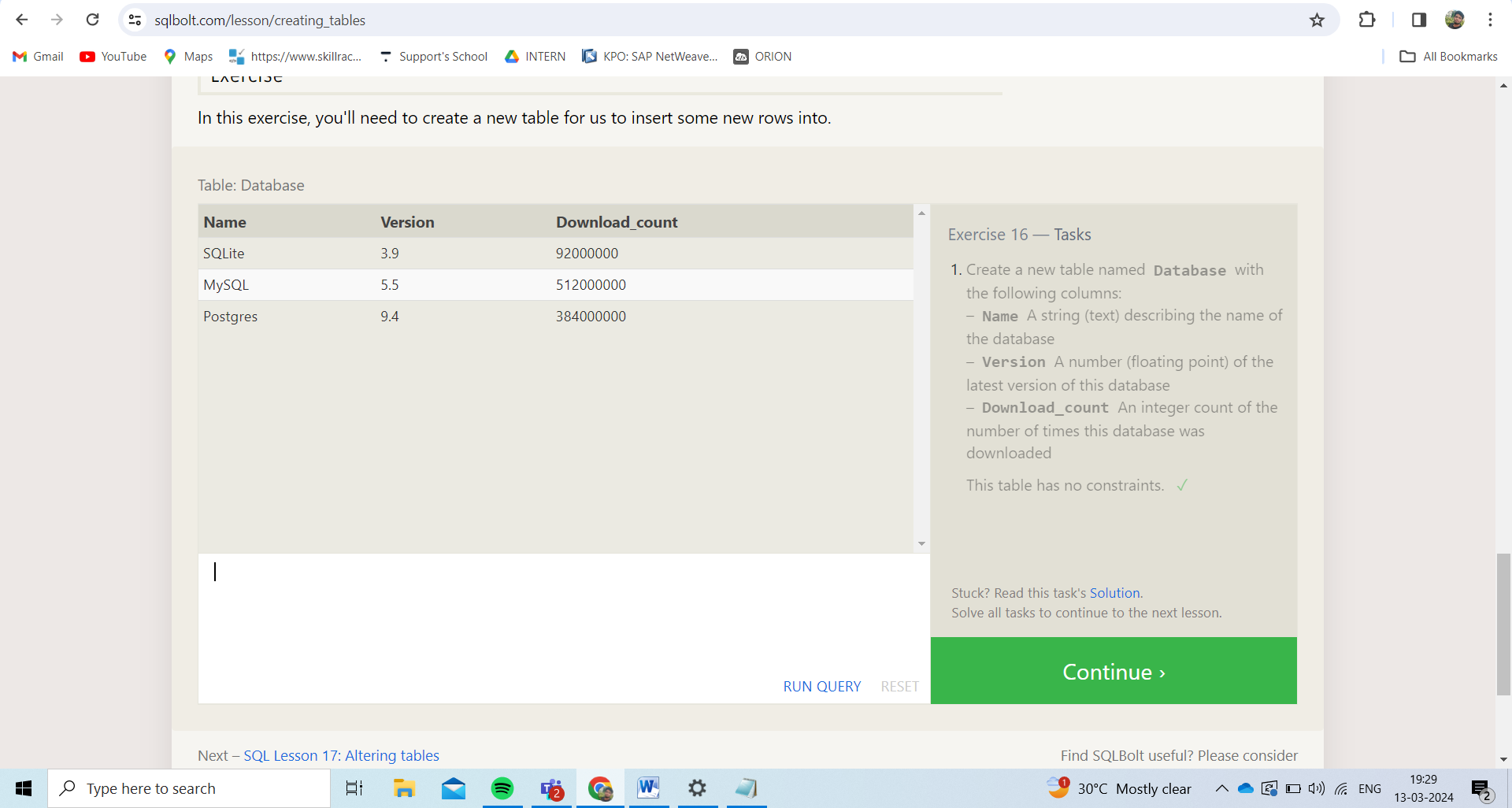
**SQL Lesson 15: Deleting rows:**



1.Delete from movies where year<2005;

2. Delete from movies where director ='Andrew Stanton';

**SQL Lesson 16: Creating tables:**



1. create table Database (

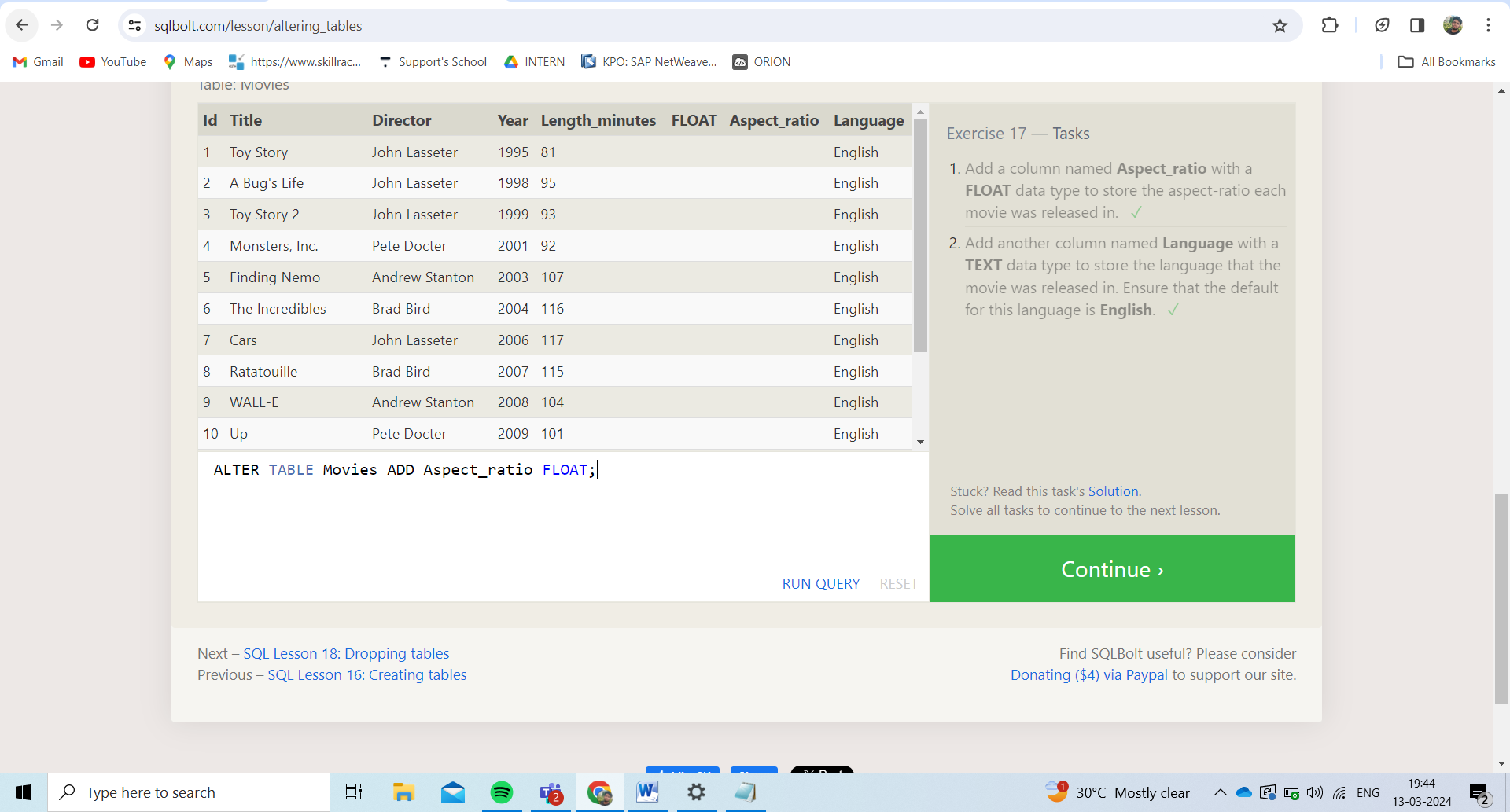
Name text,

Version float,

Download\_count int

);

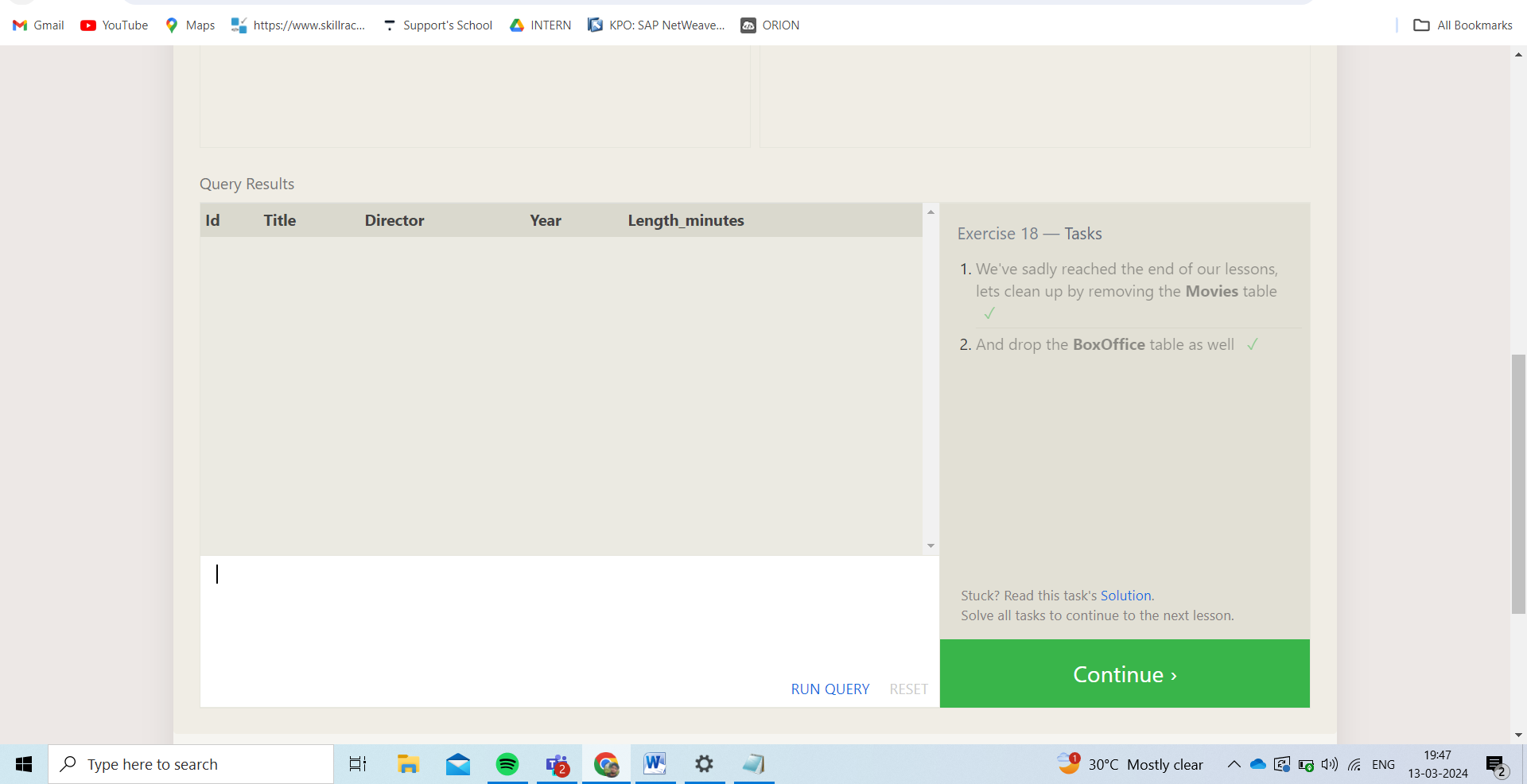
**SQL Lesson 17: Altering tables:**



1. ALTER TABLE Movies ADD Aspect\_ratio FLOAT;

2. ALTER TABLE Movies ADD Language Text default "English";

**SQL Lesson 18: Dropping tables:**



1. DROP TABLE IF EXISTS MOVIES ;
2. DROP TABLE IF EXISTS BOXOFFICE ;