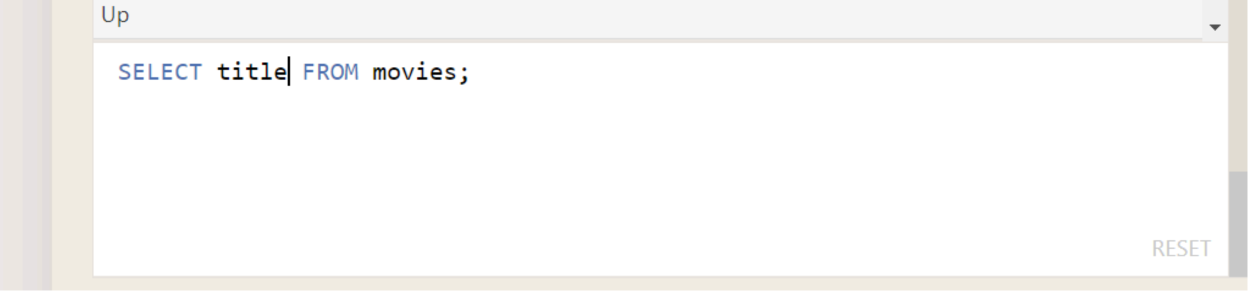
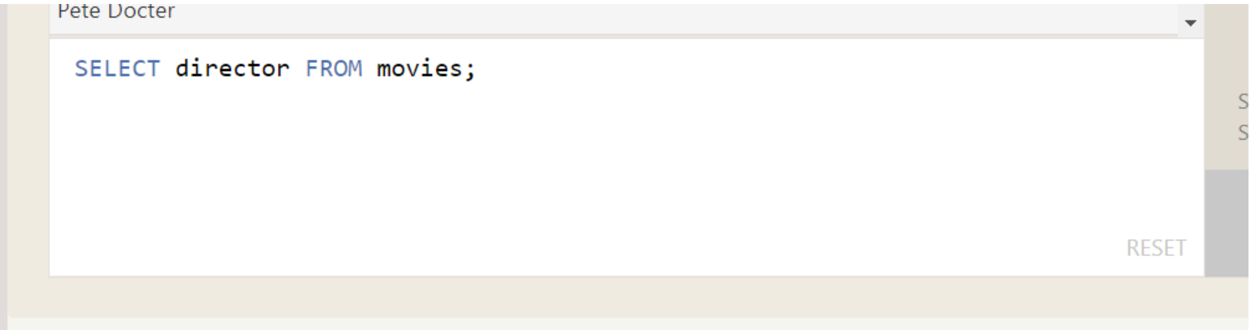
SQL Lesson 1: SELECT queries 101

1. Find the title of each film



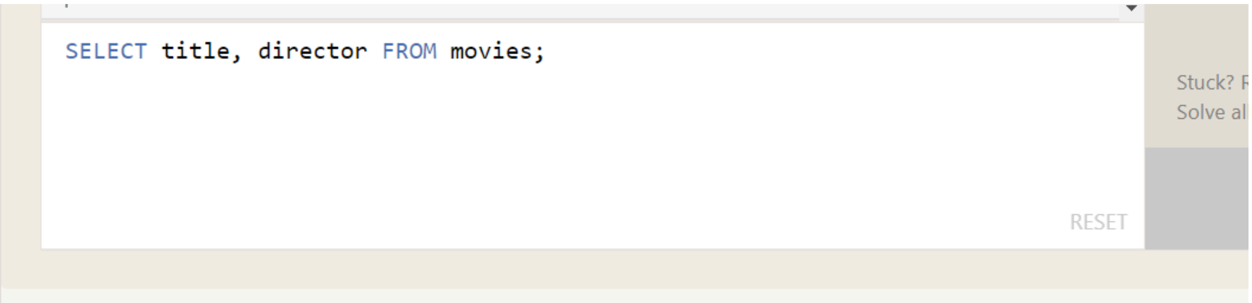
```sql
SELECT title FROM movies;
```

Find the director of each film



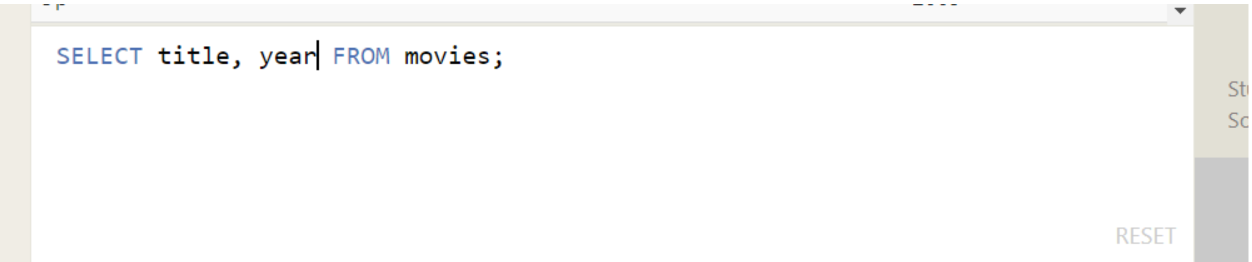
```sql
SELECT director FROM movies;
```

Find the title and director of each film



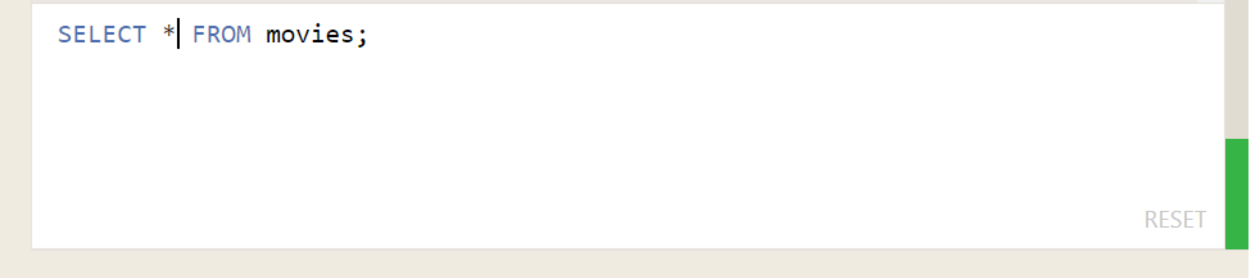
```sql
SELECT title, director FROM movies;
```

Find the title and year of each film



```
SELECT title, year FROM movies;
```

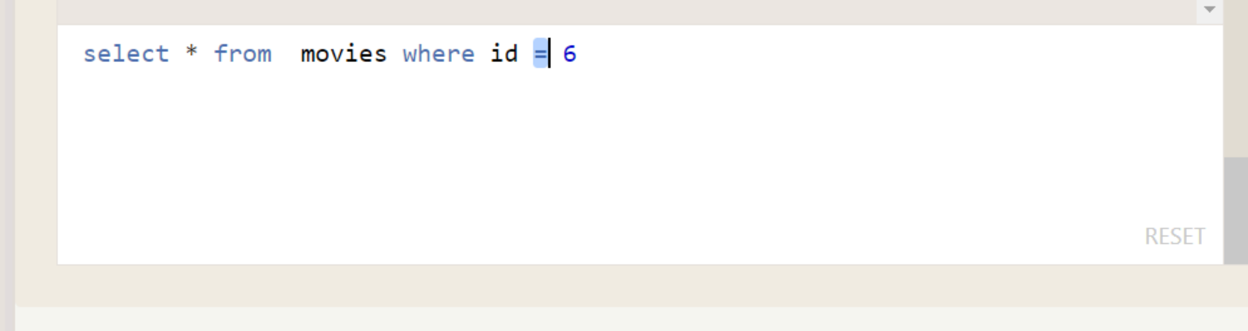
Find all the information about each film



```
SELECT * FROM movies;
```

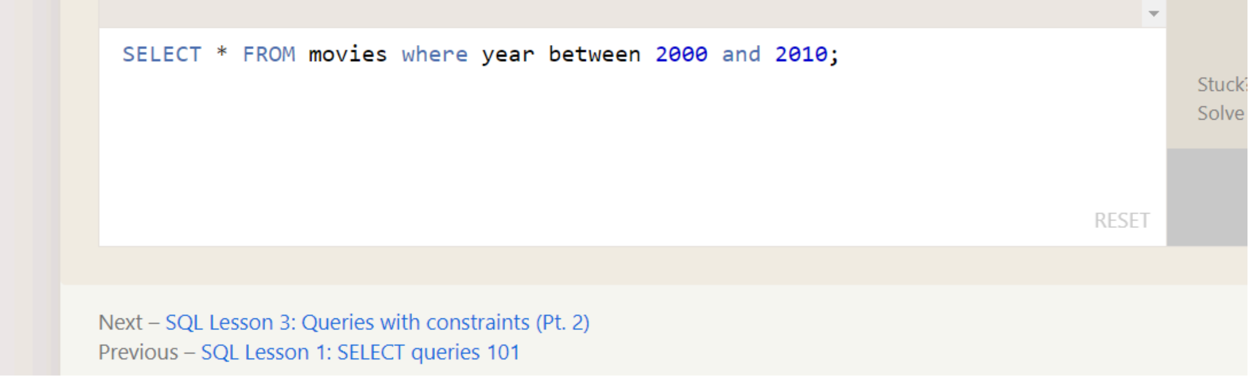
SQL Lesson 2: Queries with constraints (Pt. 1)

Find the movie with a row id of 6 ✓



```
select * from  movies where id = 6
```

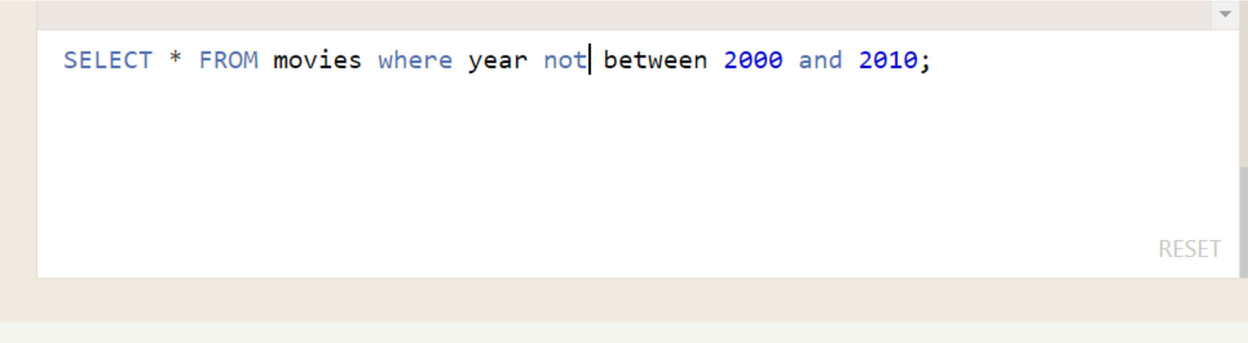
Find the movies released in the years between 2000 and 2010



```
SELECT * FROM movies where year between 2000 and 2010;
```

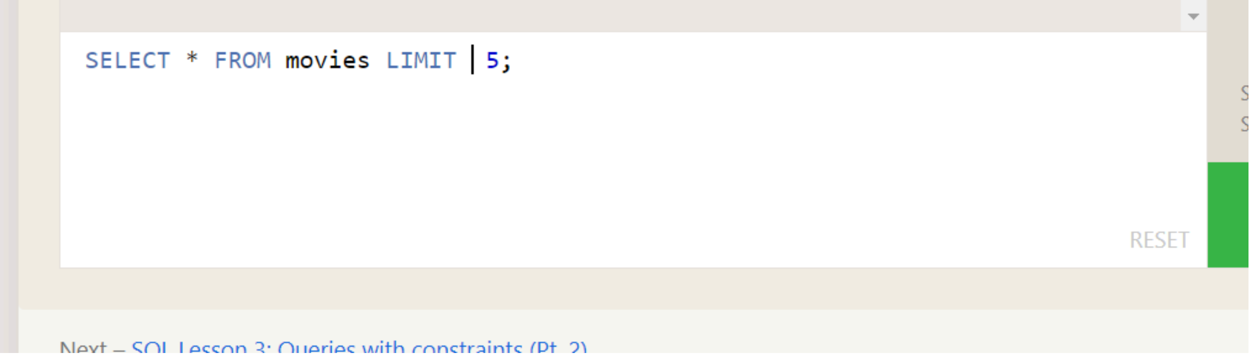
Next – SQL Lesson 3: Queries with constraints (Pt. 2)
Previous – SQL Lesson 1: SELECT queries 101

Find the movies not released in the years between 2000 and 2010



```
SELECT * FROM movies where year not between 2000 and 2010;
```

Find the first 5 Pixar movies and their release `year`

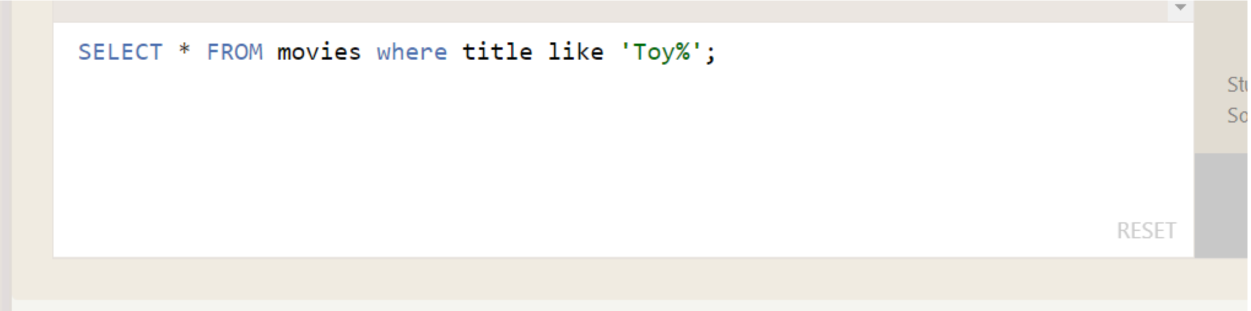


```
SELECT * FROM movies LIMIT 5;
```

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

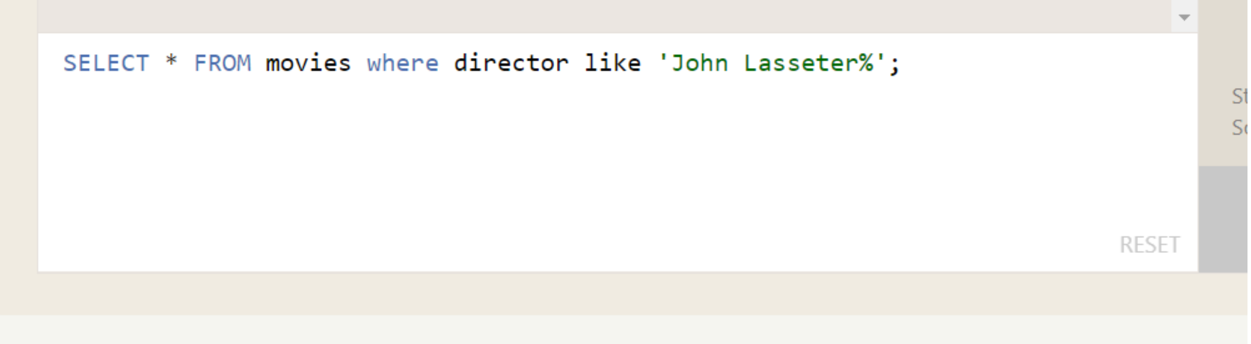
SQL Lesson 3: Queries with constraints (Pt. 2)

Find all the Toy Story movies ✓



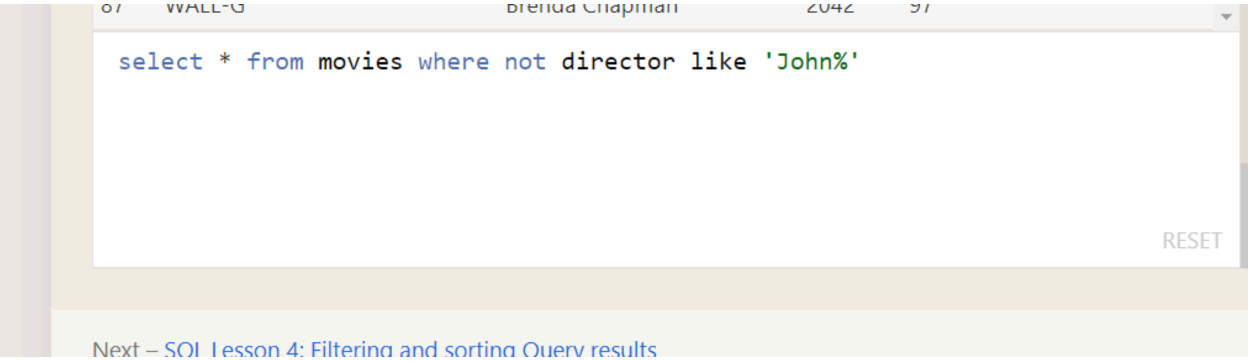
```
SELECT * FROM movies where title like 'Toy%';
```

Find all the movies directed by John Lasseter ✓



```
SELECT * FROM movies where director like 'John Lasseter%';
```

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

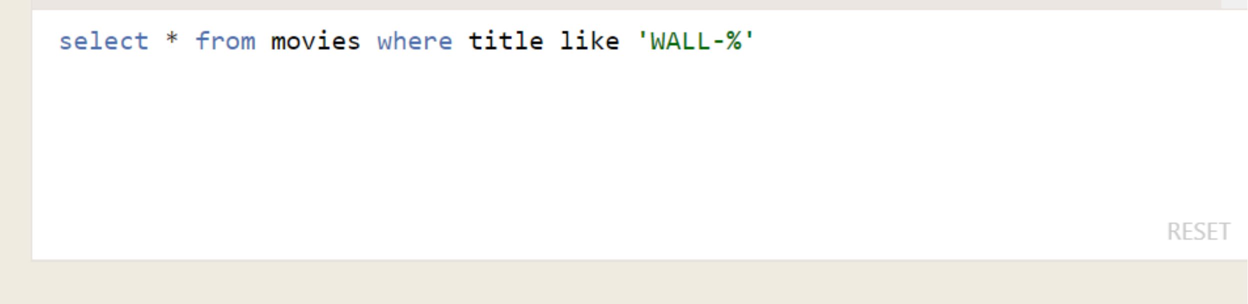


```
87    WALL-G    Brenda Chapman    2042    97
```

```
select * from movies where not director like 'John%'
```

Next – SQL Lesson 4: Filtering and sorting Query results

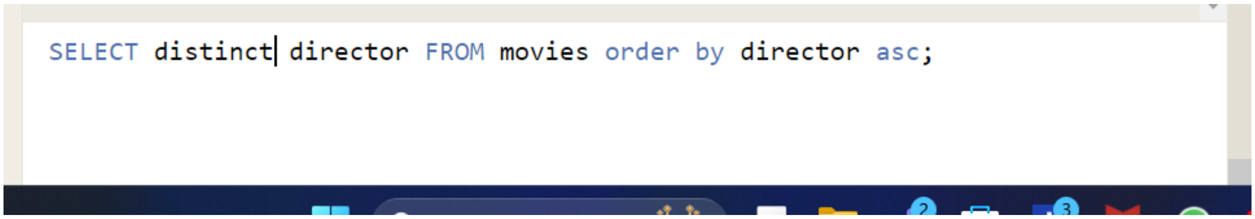
Find all the WALL-* movies



```
select * from movies where title like 'WALL-%'
```

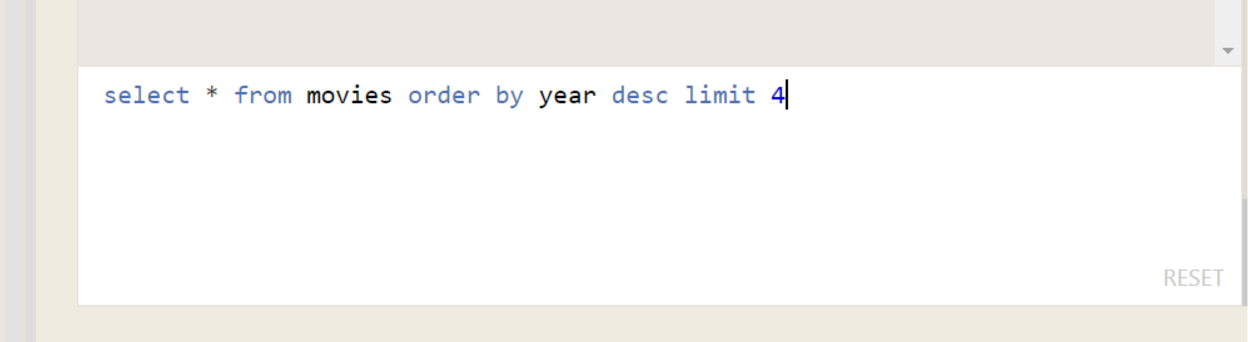
SQL Lesson 4: Filtering and sorting Query results

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



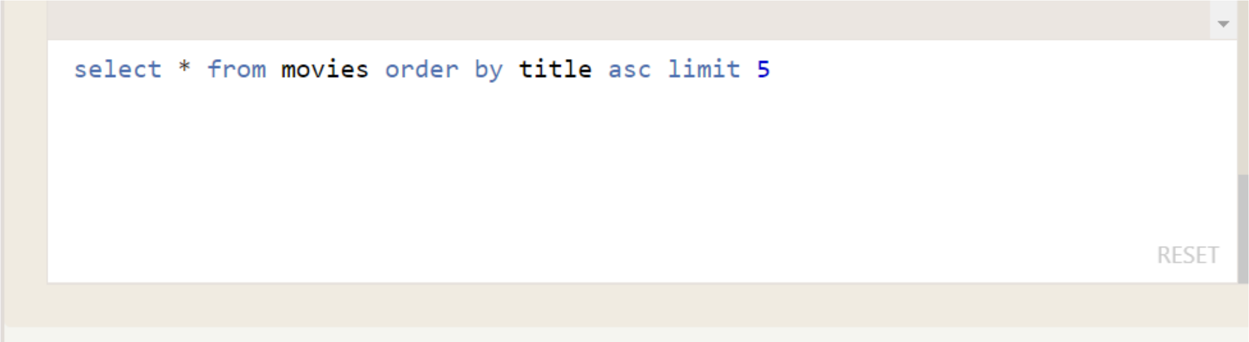
```
SELECT distinct director FROM movies order by director asc;
```

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



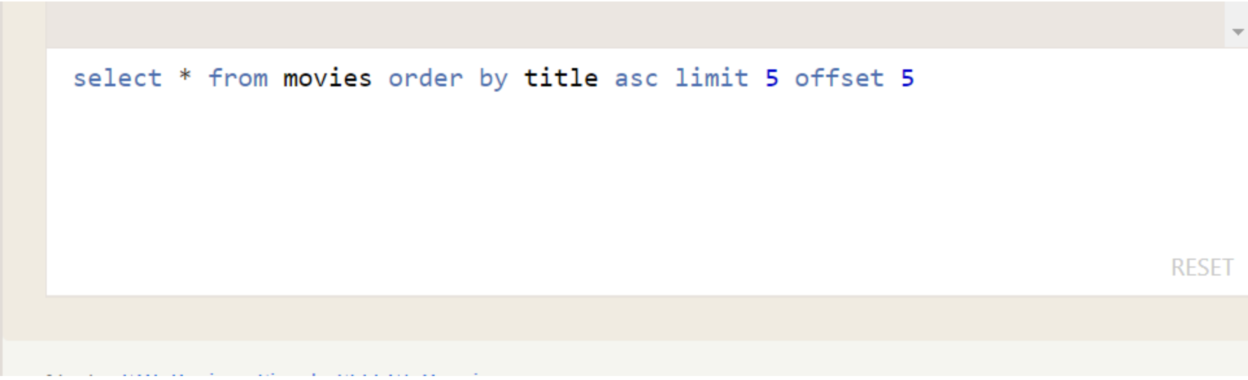
```
select * from movies order by year desc limit 4
```

List the first five Pixar movies sorted alphabetically



```
select * from movies order by title asc limit 5
```

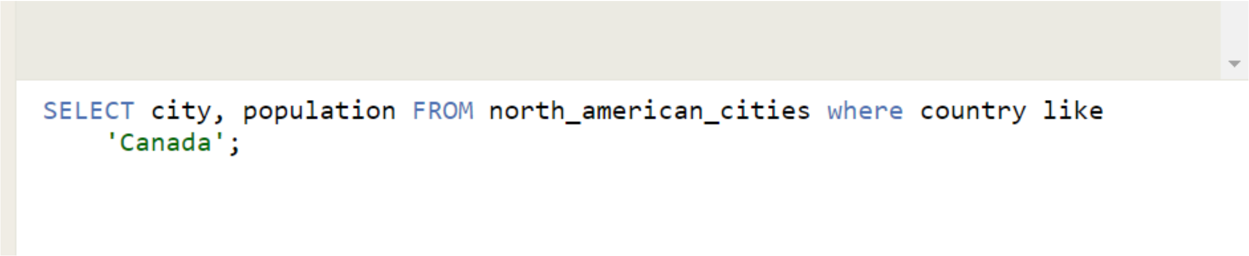
List the next five Pixar movies sorted alphabetically



```
select * from movies order by title asc limit 5 offset 5
```

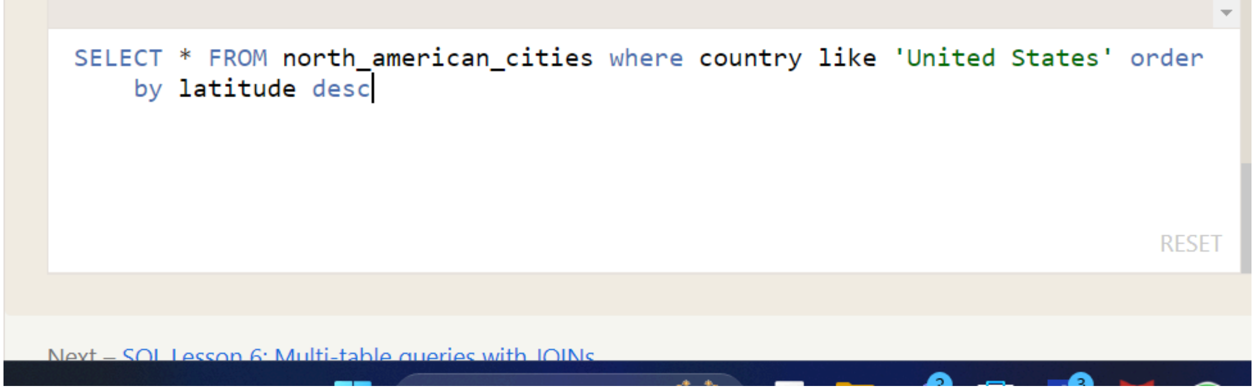
SQL Review: Simple SELECT Queries

List all the Canadian cities and their populations



```
SELECT city, population FROM north_american_cities where country like
    'Canada';
```

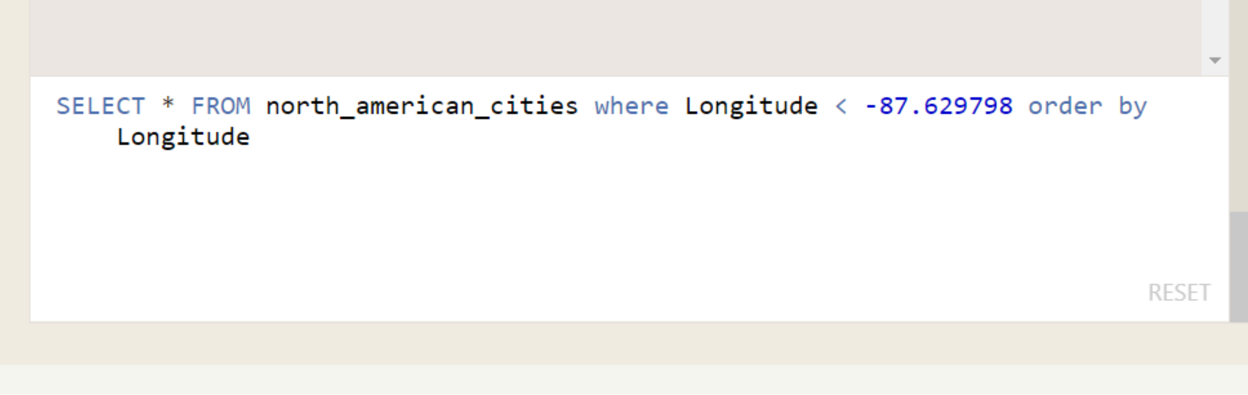
Order all the cities in the United States by their latitude from north to south



```
SELECT * FROM north_american_cities where country like 'United States' order
    by latitude desc
```

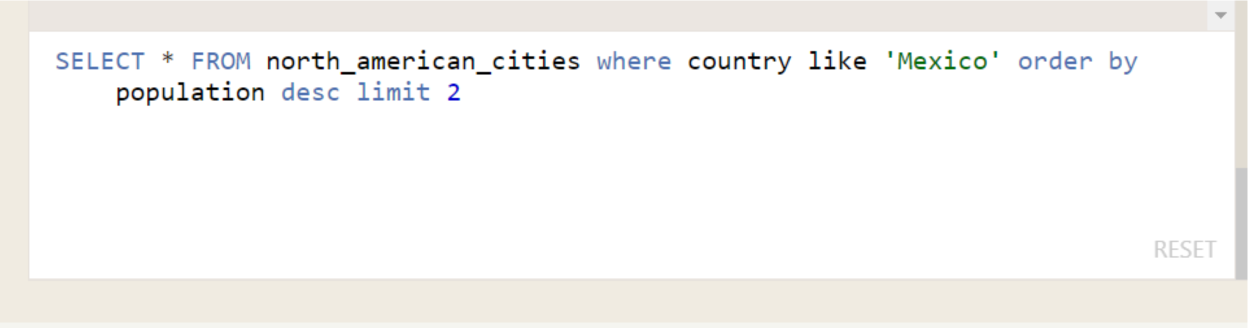
Next – SQL Lesson 6: Multi-table queries with JOINs

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



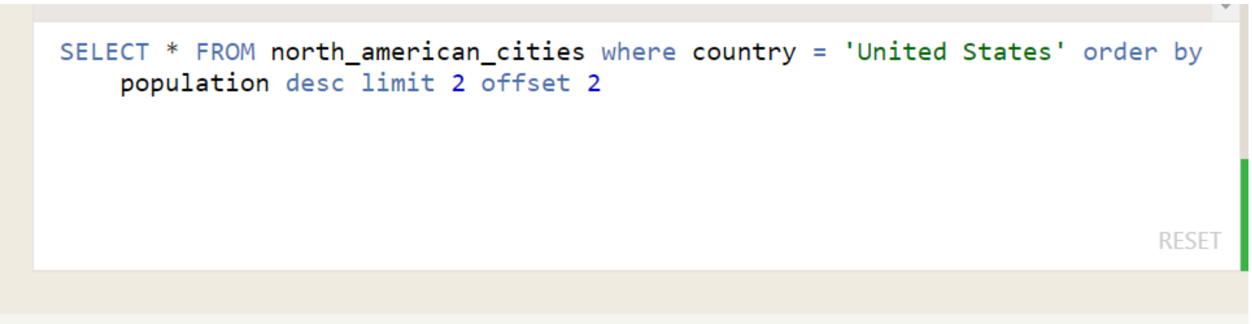
```
SELECT * FROM north_american_cities where Longitude < -87.629798 order by
    Longitude
```

List the two largest cities in Mexico (by population)



```
SELECT * FROM north_american_cities where country like 'Mexico' order by
    population desc limit 2
```

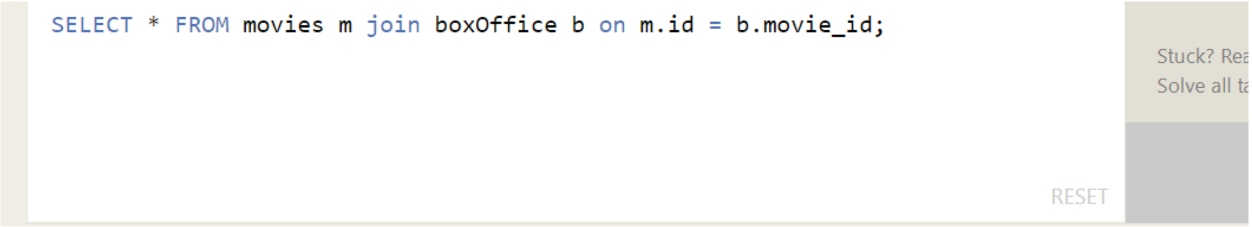
List the third and fourth largest cities (by population) in the United States and their population



```
SELECT * FROM north_american_cities where country = 'United States' order by
    population desc limit 2 offset 2
```

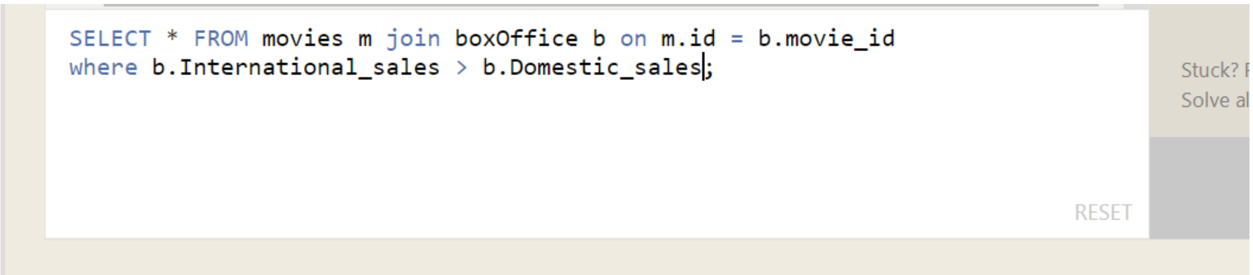
SQL Lesson 6: Multi-table queries with JOINs

Find the domestic and international sales for each movie



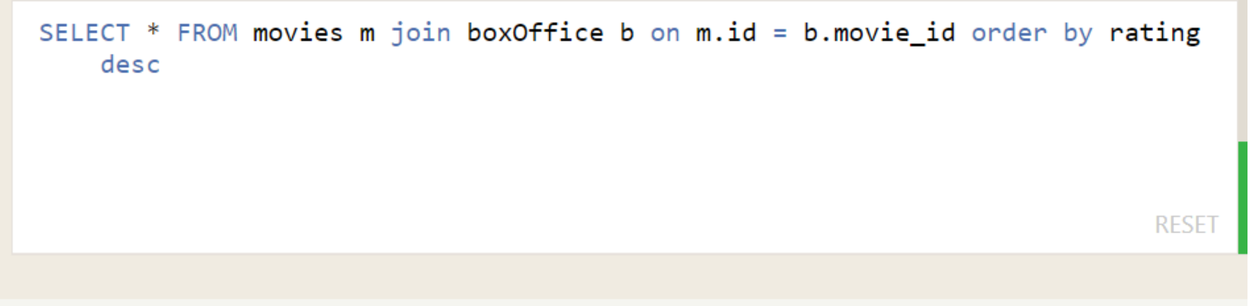
```
SELECT * FROM movies m join boxOffice b on m.id = b.movie_id;
```

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



```
SELECT * FROM movies m join boxOffice b on m.id = b.movie_id
where b.International_sales > b.Domestic_sales;
```

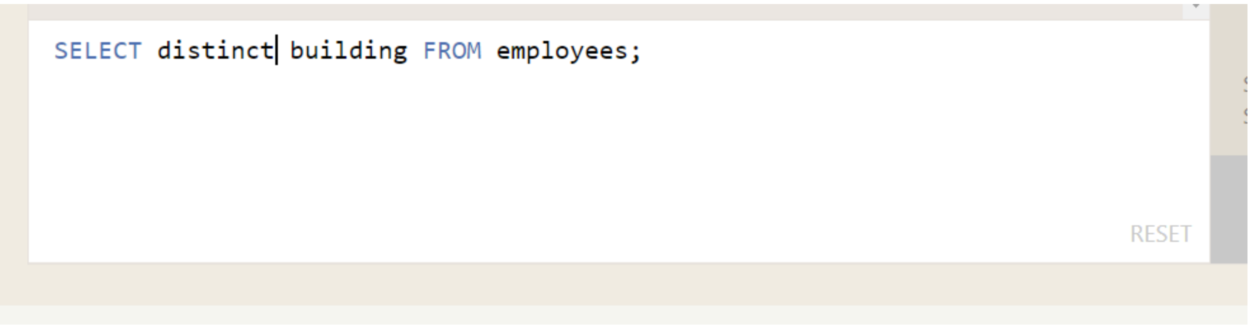
List all the movies by their ratings in descending order



```
SELECT * FROM movies m join boxOffice b on m.id = b.movie_id order by rating
    desc
```

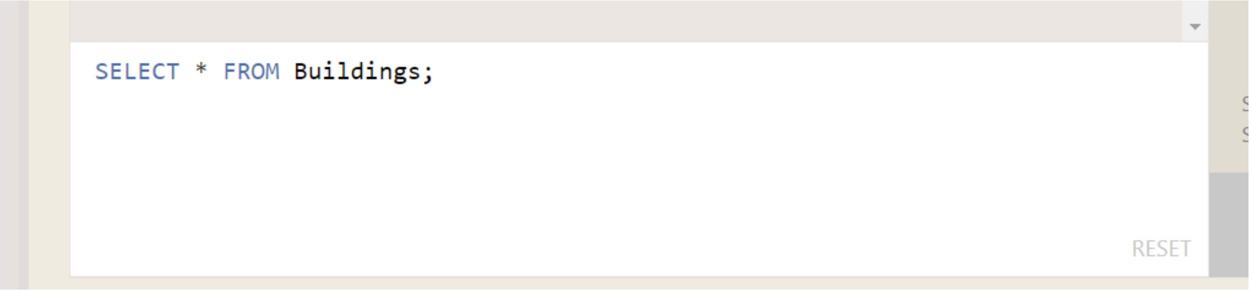
SQL Lesson 7: OUTER JOINs

Find the list of all buildings that have employees



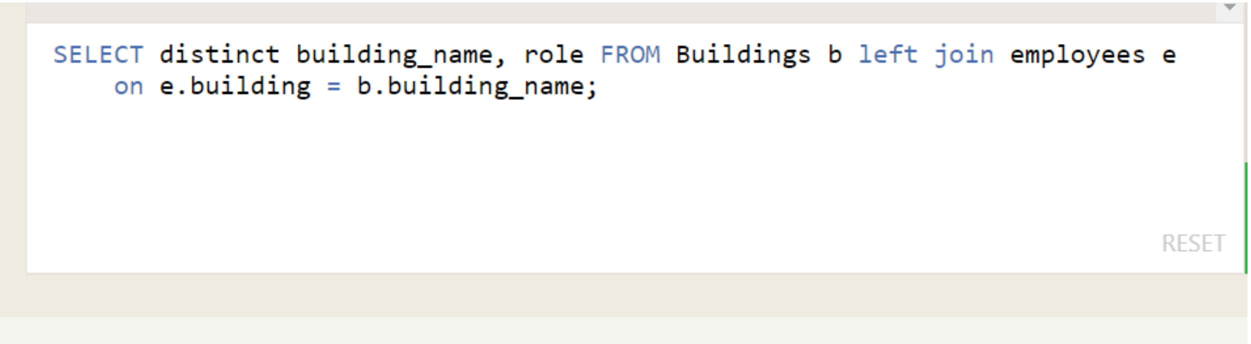
```
SELECT distinct building FROM employees;
```

Find the list of all buildings and their capacity



```
SELECT * FROM Buildings;
```

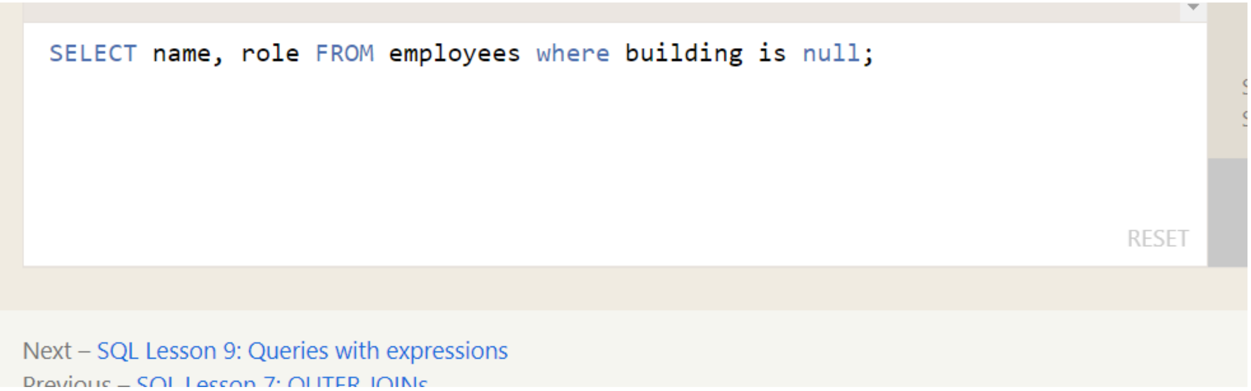
List all buildings and the distinct employee roles in each building (including empty buildings



```
SELECT distinct building_name, role FROM Buildings b left join employees e
    on e.building = b.building_name;
```

SQL Lesson 8: A short note on NULLs

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

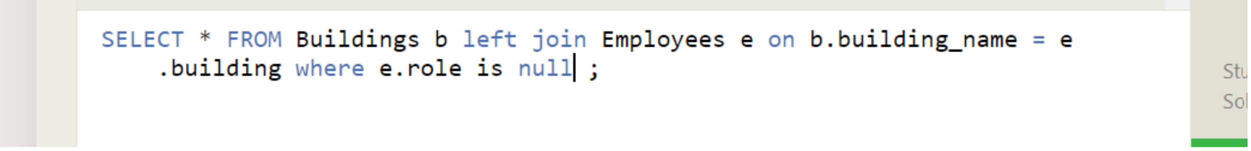


```
SELECT name, role FROM employees where building is null;
```

Next – SQL Lesson 9: Queries with expressions

Previous – SQL Lesson 7: OUTER JOINs

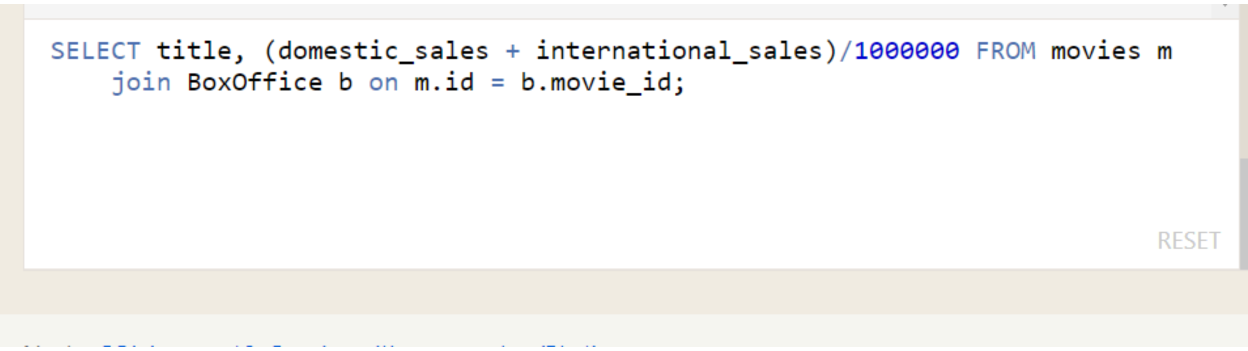
Find the names of the buildings that hold no employees



```
SELECT * FROM Buildings b left join Employees e on b.building_name = e
    .building where e.role is null ;
```

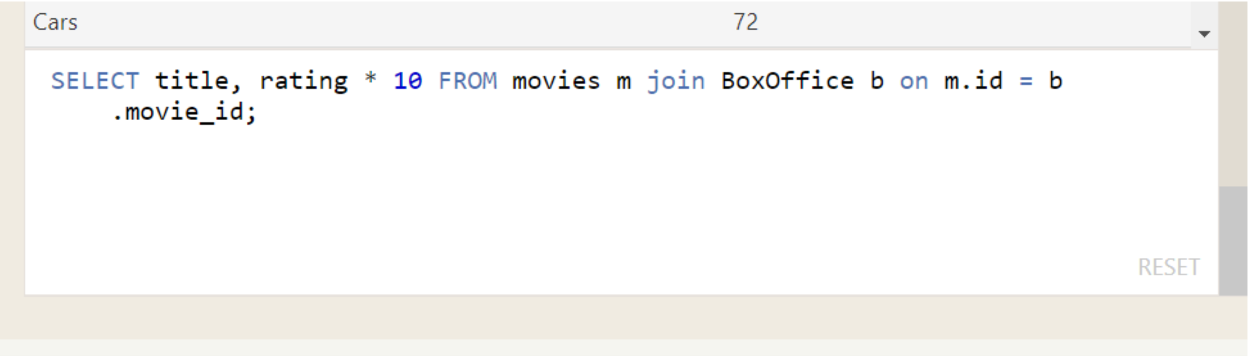
SQL Lesson 9: Queries with expressions

List all movies and their combined sales in millions of dollars ✓



```
SELECT title, (domestic_sales + international_sales)/1000000 FROM movies m
    join BoxOffice b on m.id = b.movie_id;
```

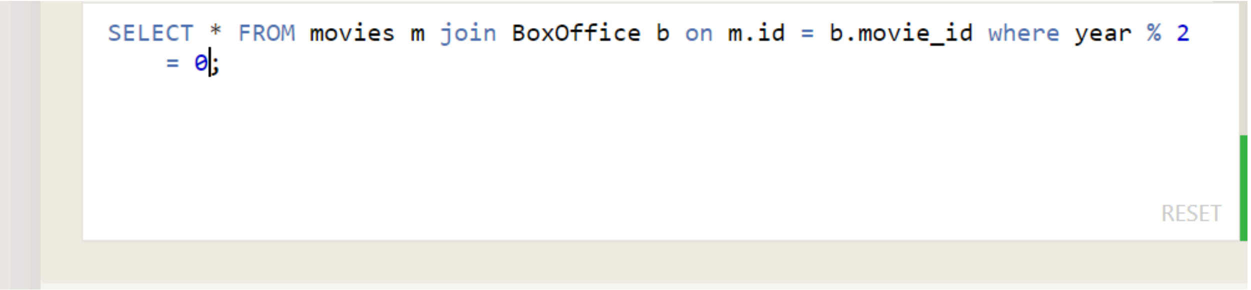
List all movies and their ratings in percent



```
Cars                                            72
```

```
SELECT title, rating * 10 FROM movies m join BoxOffice b on m.id = b
    .movie_id;
```

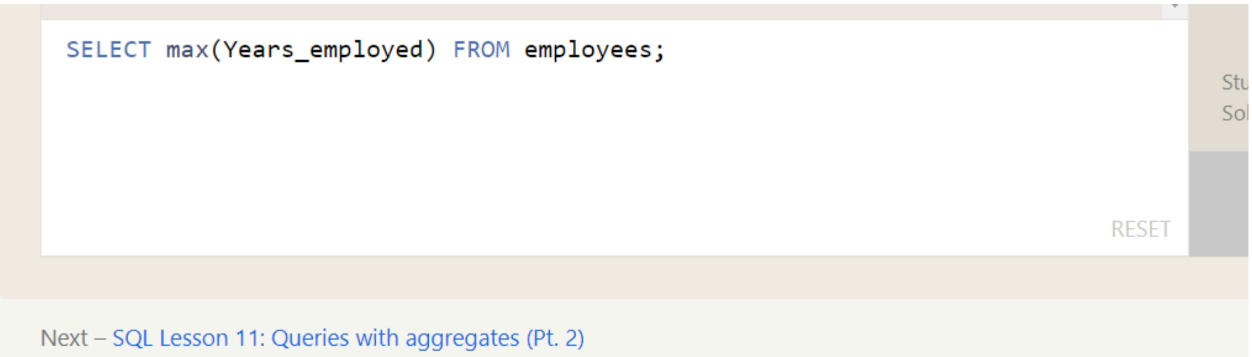
List all movies that were released on even number years



```
SELECT * FROM movies m join BoxOffice b on m.id = b.movie_id where year % 2
    = 0;
```

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

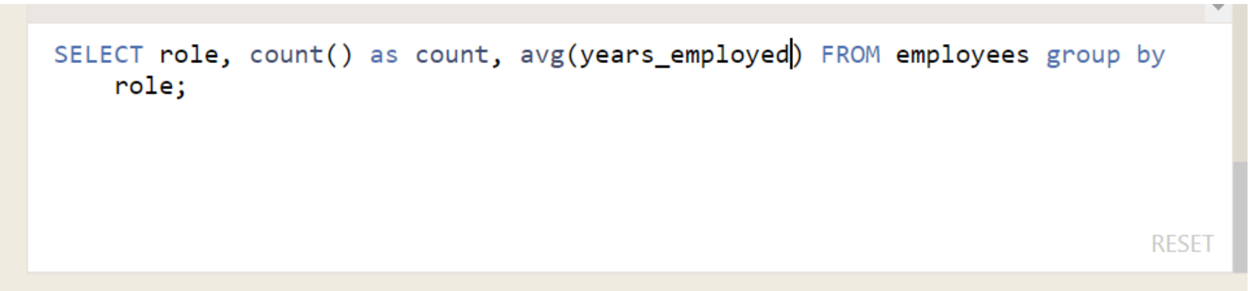
Find the longest time that an employee has been at the studio



```
SELECT max(Years_employed) FROM employees;
```

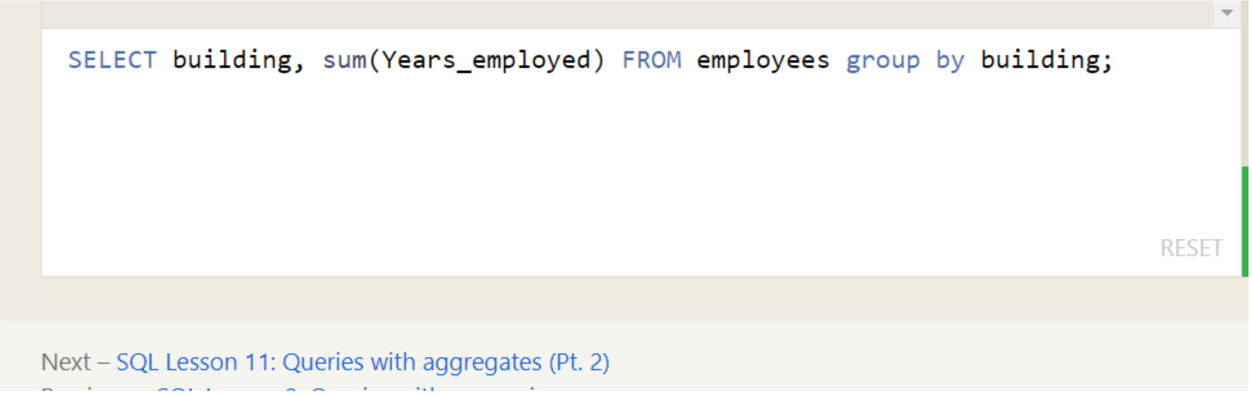
Next – SQL Lesson 11: Queries with aggregates (Pt. 2)

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



```
SELECT role, count() as count, avg(years_employed) FROM employees group by
    role;
```

Find the total number of employee years worked in each building

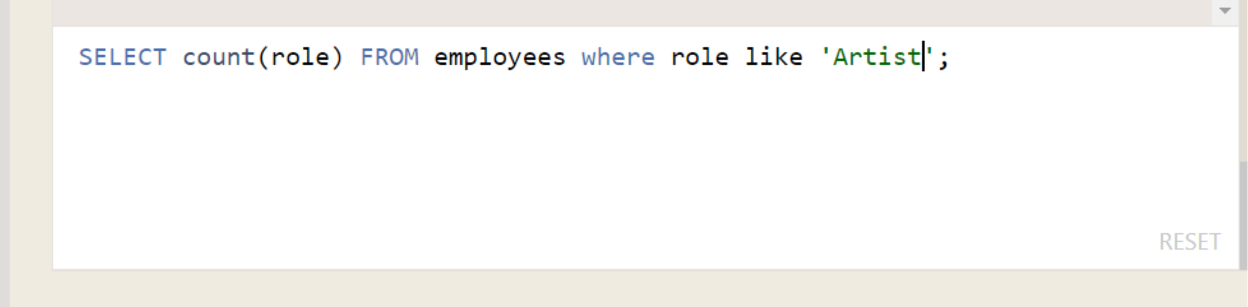


```
SELECT building, sum(Years_employed) FROM employees group by building;
```

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

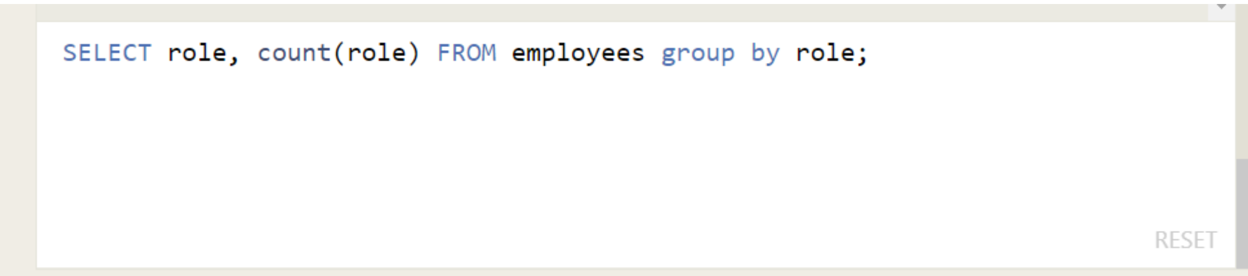
SQL Lesson 11: Queries with aggregates (Pt. 2)

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



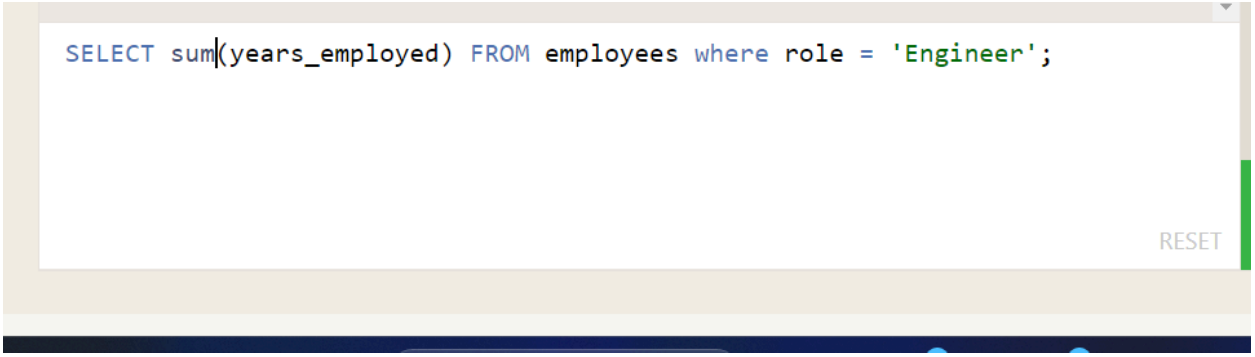
```
SELECT count(role) FROM employees where role like 'Artist';
```

Find the number of Employees of each role in the studio



```
SELECT role, count(role) FROM employees group by role;
```

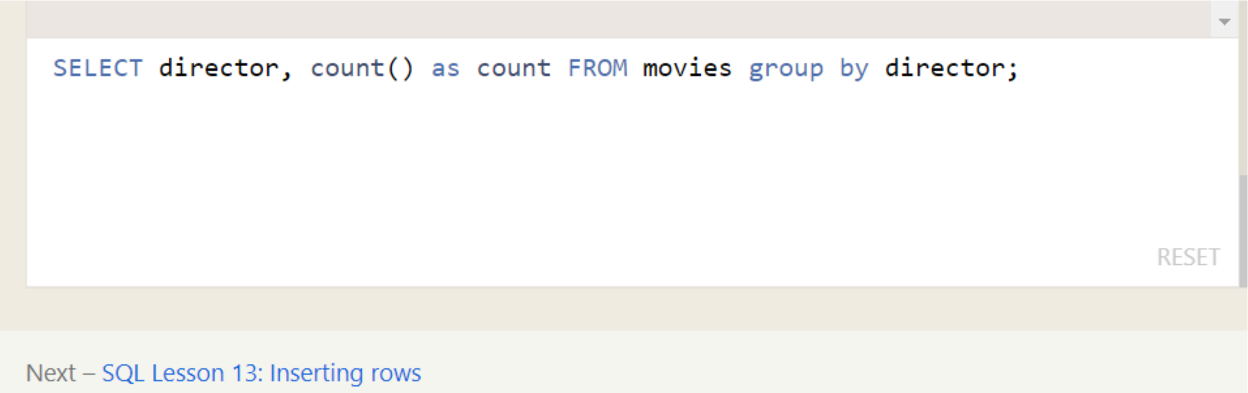
Find the total number of years employed by all Engineers



```
SELECT sum(years_employed) FROM employees where role = 'Engineer';
```

SQL Lesson 12: Order of execution of a Query

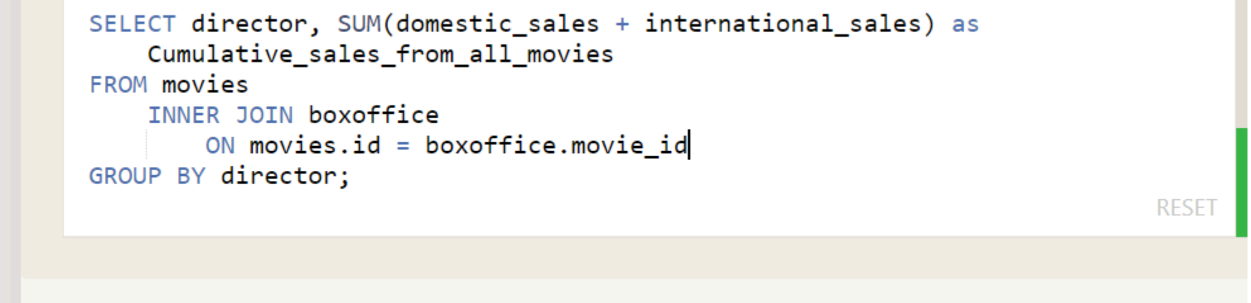
Find the number of movies each director has directed



```
SELECT director, count() as count FROM movies group by director;
```

Next – SQL Lesson 13: Inserting rows

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



```
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;
```