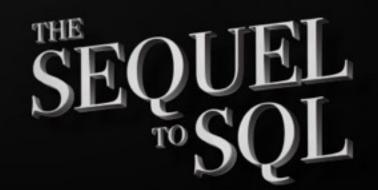
The Sequel to SQL: Level 4 – Section 2 Aliases



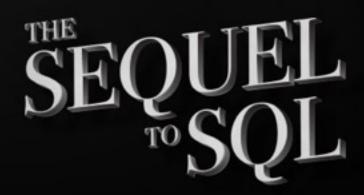
Giving Our Results More Meaningful Names

Can we give our columns a name that has a more accurate meaning?

The Current Query

```
SELECT Movies.title , Reviews.review
FROM Movies
INNER JOIN Reviews
ON Movies.id=Reviews.movie_id;
```

title	review
Don Juan	Loved it! A must-see! Hated it It was okay Do not see!



Using Column Aliases

Give the columns new temporary names.

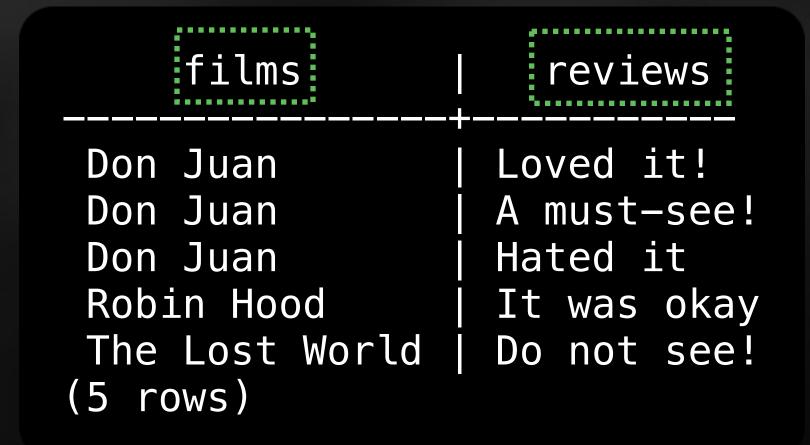
SELECT Movies.title AS films, Reviews.review AS reviews

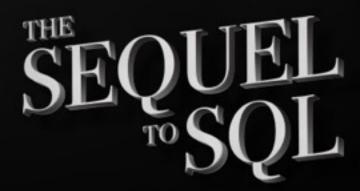
FROM Movies

INNER JOIN Reviews

ON Movies.id=Reviews.movie_id;

These are temporary table names that will only affect this query.



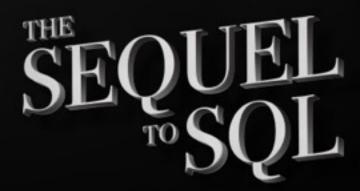


Using Column Aliases

The AS can be dropped from the query.

```
SELECT Movies.title films, Reviews.review reviews
FROM Movies
INNER JOIN Reviews
ON Movies.id=Reviews.movie_id;
```

```
Don Juan | Loved it!
Don Juan | A must-see!
Don Juan | Hated it
Robin Hood | It was okay
The Lost World | Do not see!
(5 rows)
```



Using More Than 1 Word for a Column Alias

SELECT Movies.title "Weekly Movies",
Reviews.review "Weekly Reviews"
FROM Movies
INNER JOIN Reviews
ON Movies.id=Reviews.movie_id;

Weekly Films | Weekly Reviews

Don Juan | Loved it!

Don Juan | A must-see!

Don Juan | Hated it

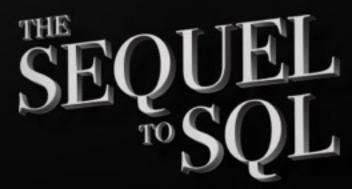
Robin Hood | It was okay

The Lost World | Do not see!

(5 rows)

When using aliases with more than 2 words, you must use quotation marks.

Quotes are also needed if you want capitalization.



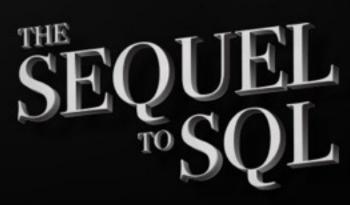
Our Queries Can Get Verbose

Every time we need to reference the Movies or Promotions table, we have to type the whole word.

Our Original Query

```
SELECT Movies.title, Reviews.review
FROM Movies
INNER JOIN Reviews
ON Movies.id=Reviews.movie_id
ORDER BY Movies.title;
```

By using **Table Aliases**, we can shorten this query by substituting the Table Name.



Using Table Aliases

By shortening our queries, we can save time when producing longer queries.

```
SELECT m.title, Reviews.review
FROM Movies m
INNER JOIN Reviews
ON m.id=Reviews.movie_id
ORDER BY m.title;
```

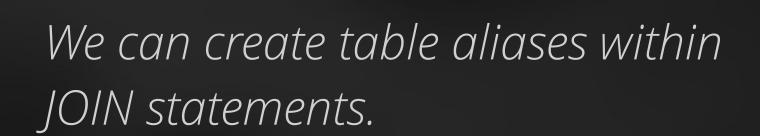
Allows us to refer to the Movies table as m



Using Table Aliases

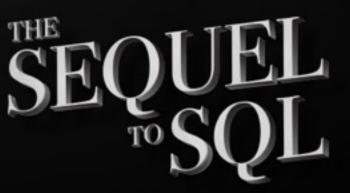
Table aliases can also be created on any or all tables both within and outside the FROM clause.

SELECT m.title, r.review
FROM Movies m
INNER JOIN Reviews r
ON m.id=r.movie_id
ORDER BY m.title;



title	review
Don Juan Don Juan	Loved it! A must-see! Hated it It was okay Do not see!

Same result!



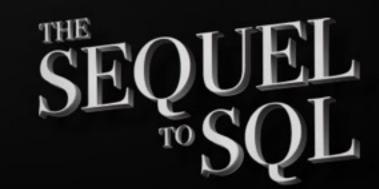
Remember This INNER JOIN?

How do we find the genres of *Peter Pan*?

```
SELECT Movies.title, Genres.name
FROM Movies

INNER JOIN Movies_Genres
ON Movies.id = Movies_Genres.movie_id

INNER JOIN Genres
ON Movies_Genres.id = Genres.id
WHERE Movies.title = "Peter Pan";
```



INNER JOIN on Multiple Tables With Aliases

How do we find the genres of *Peter Pan*?

SELECT m.title, g.name FROM Movies m First join - INNER JOIN Movies_Genres mg $ON m.id = mg.movie_id$ Second join - INNER JOIN Genres g ON mg.genre_id = g.id WHERE m.title = "Peter Pan";

