Creating a table in PostgreSQL

Columns will be user_id, name, email and username

User_id will be assigned automatically – SQL will assign a unique number that increments with each INSERT. This is what SERIAL means, and using it as the PRIMARY KEY means that's the column that will always have a value that's unique to each row.

We are also setting the data type of name, email and username so they must be text. (You MUST declare these data types otherwise you will get errors)

Instead of SERIAL, you may see an Identity Column instead (introduced in PostgreSQL version 10). This is because you can accidentally UPDATE info in the column is SERIAL!

To specify identity column, specify INT PRIMARY KEY GENERATED ALWAYS AS IDENTITY. These values cannot be overwritten using UPDATE (but can still be deleted as normal).

```
CREATE TABLE users (
    user_id INT PRIMARY KEY GENERATED
ALWAYS AS IDENTITY,
    name TEXT,
    email TEXT,
    username TEXT
);
```

SQL JOINS

Examples use: https://www.db-fiddle.com/f/4FJEKAFU4SS5uECGdLeXgM/0

Allow us to combine data from 2 different tables based on a related column.
Why is this useful? Real world example - think about a filing cabinet – if we had a table containing people and a table containing their documents, we could use JOIN to find all the documents belonging to a particular person

Using a join is much quicker and more efficient than doing 2 separate queries.

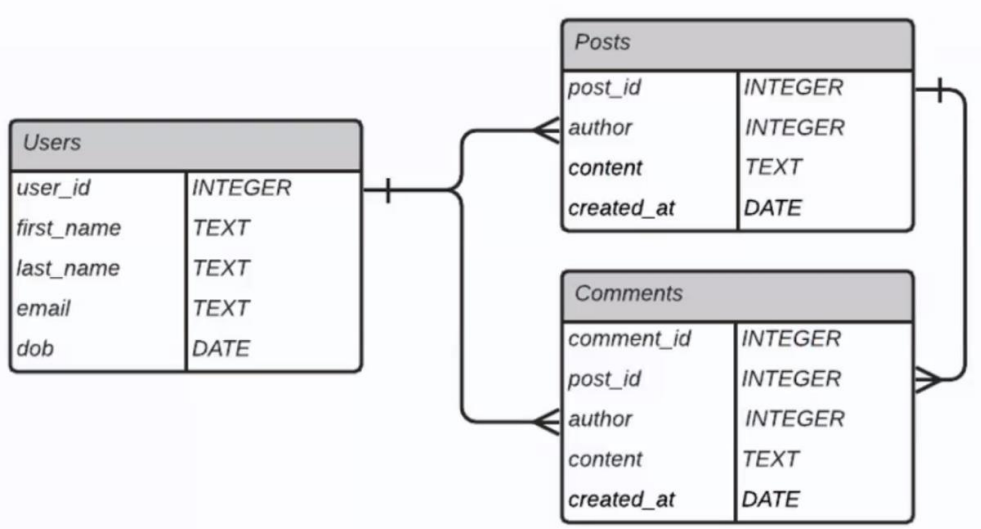
Why not keep all the data in one table? Reduces duplication, reduces layers of complexity, keeps the data secure (e.g. people can only access what they need), reducing documentation (e.g. only updating data in one place), reducing nulls.

Storing data across multiple tables is called normalisation. Normalisation allows us to keep data redundancy low so that we can decrease the amount of data anomalies in our application, especially when we delete or update a record.

Entity relationship diagram (ERD)

Describes what is called the Schema (tables)

The below example is for a blog website:



The lines are showing that the user_id in Users is being used as the related column to the author in Comments and Posts, the author will be stored as user_id. Post_id from Posts can be used for post_id to link Comments to Posts.
The crows feet signifies "one to many" – one author (user_id) could have written several posts and comments. One post_id may have several comments

Inner Join

The most common type of join. Similar to Venn diagrams – things which overlap!

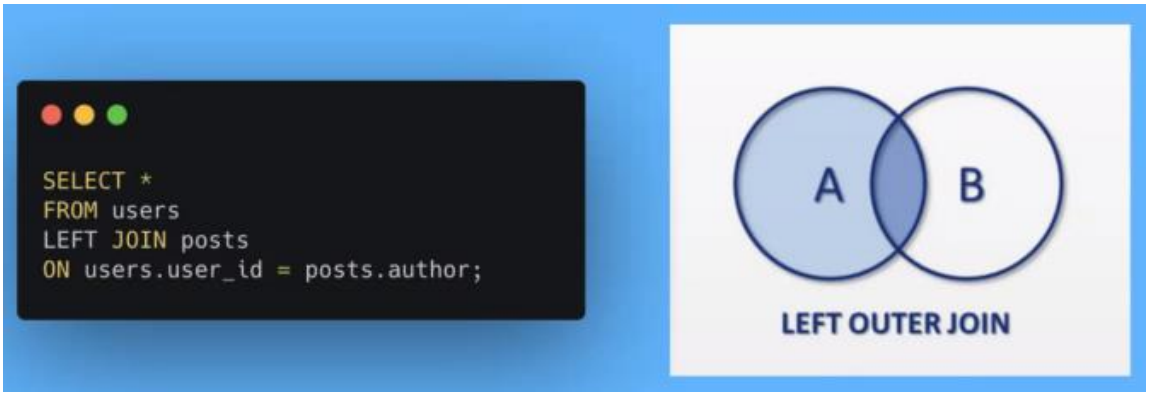
Only returns data where the 2 tables overlap.
If you see "JOIN", then it automatically defaults to INNER JOIN (for your reference), but you should always specify as best practice!

```
SELECT *
FROM users
INNER JOIN posts
ON users.user_id = posts.author
```

Selects all columns, from a combined table of users joined to posts, where the user_id matches author. So in our example, only 2 authors have written posts so those 2 authors are displayed with all the posts they have written.

Left Outer Join

Gets all the records from table A, whether or not they have null values and only the rows from table B where the records match.



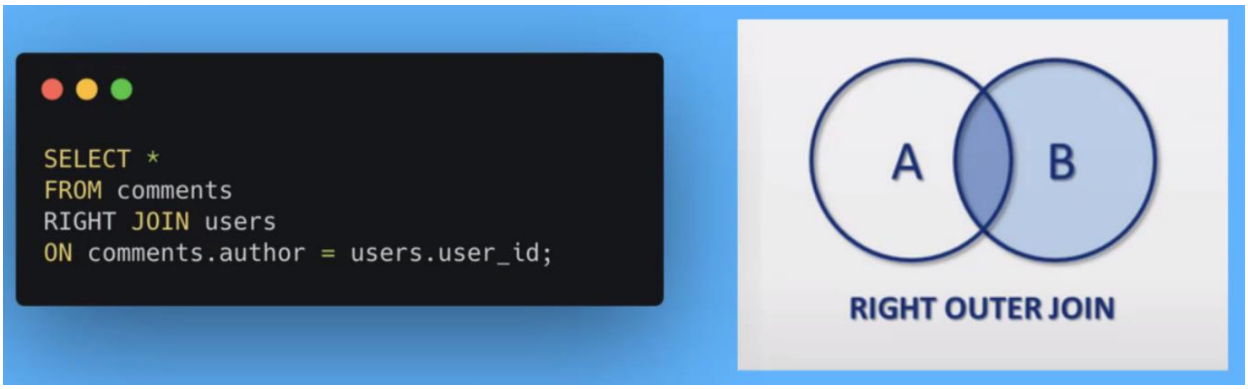

```
SELECT *
FROM users
LEFT JOIN posts
ON users.user_id = posts.author
```

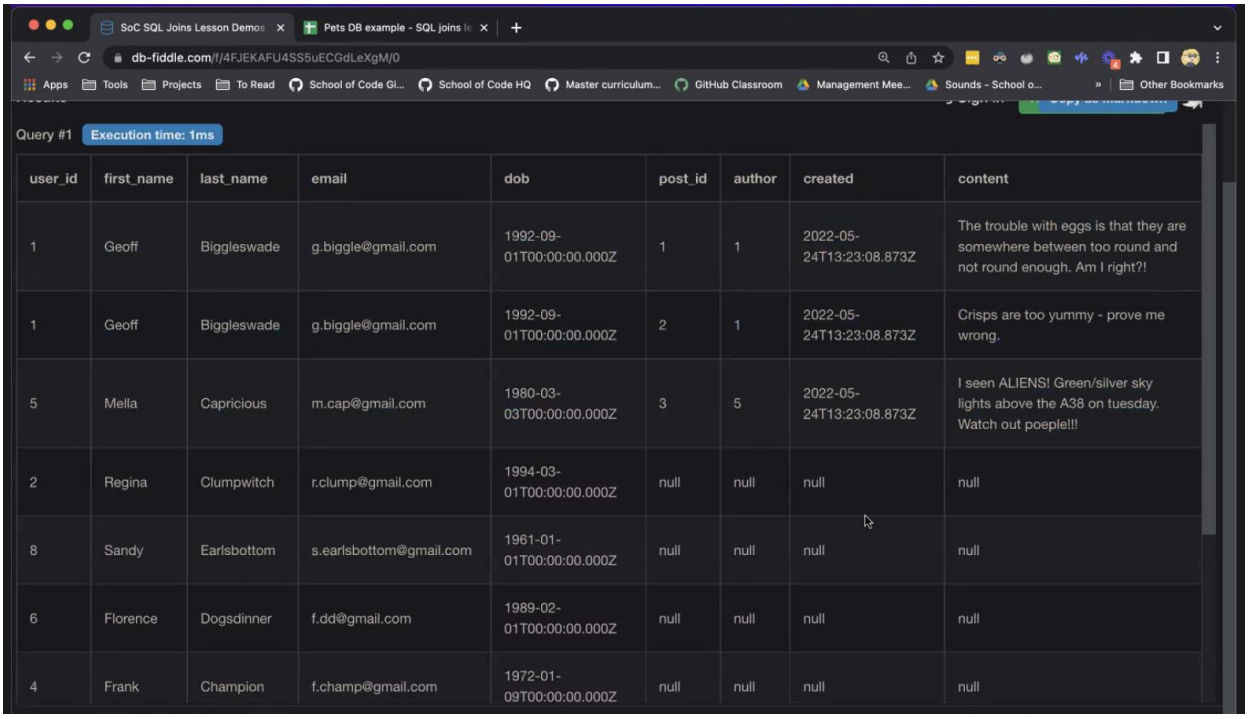
Select all columns, from a combined table that has all the users and showing posts where there are some. Blank values from table B are filled in with null.

Right Outer Join

```
SELECT *
FROM comments
RIGHT JOIN users
ON comments.author = users.user_id;
```

Returns all rows from table B, and only the rows from Table A that match the query





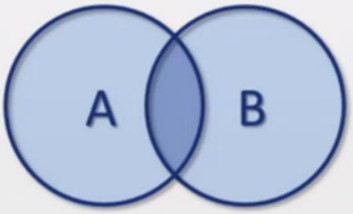
user_id	first_name	last_name	email	dob	post_id	author	created	content
1	Geoff	Biggleswade	g.biggle@gmail.com	1992-09-01T00:00:00.000Z	1	1	2022-05-24T13:23:08.873Z	The trouble with eggs is that they are somewhere between too round and not round enough. Am I right?!
1	Geoff	Biggleswade	g.biggle@gmail.com	1992-09-01T00:00:00.000Z	2	1	2022-05-24T13:23:08.873Z	Crisps are too yummy - prove me wrong.
5	Mella	Capricious	m.cap@gmail.com	1980-03-03T00:00:00.000Z	3	5	2022-05-24T13:23:08.873Z	I seen ALIENS! Green/silver sky lights above the A38 on tuesday. Watch out poeple!!!
2	Regina	Clumpwitch	r.clump@gmail.com	1994-03-01T00:00:00.000Z	null	null	null	null
8	Sandy	Earlsbottom	s.earlsbottom@gmail.com	1961-01-01T00:00:00.000Z	null	null	null	null
6	Florence	Dogsdinner	f.dd@gmail.com	1989-02-01T00:00:00.000Z	null	null	null	null
4	Frank	Champion	f.champ@gmail.com	1972-01-09T00:00:00.000Z	null	null	null	null

Full Outer Join

Quite rare.
Acts like a left join and right join at the same time. Returns everything from both tables, filling in null where things don't match up. Can get massive results with lots of nulls, so be careful!

```
SELECT *
FROM posts
FULL JOIN users
ON posts.author = users.user_id
```

```sql
SELECT *
FROM posts
FULL JOIN users
ON posts.author = users.user_id;
```



FULL OUTER JOIN