CT4009

Advanced SQL and MySQL

Andy Bell

What we’ll cover today

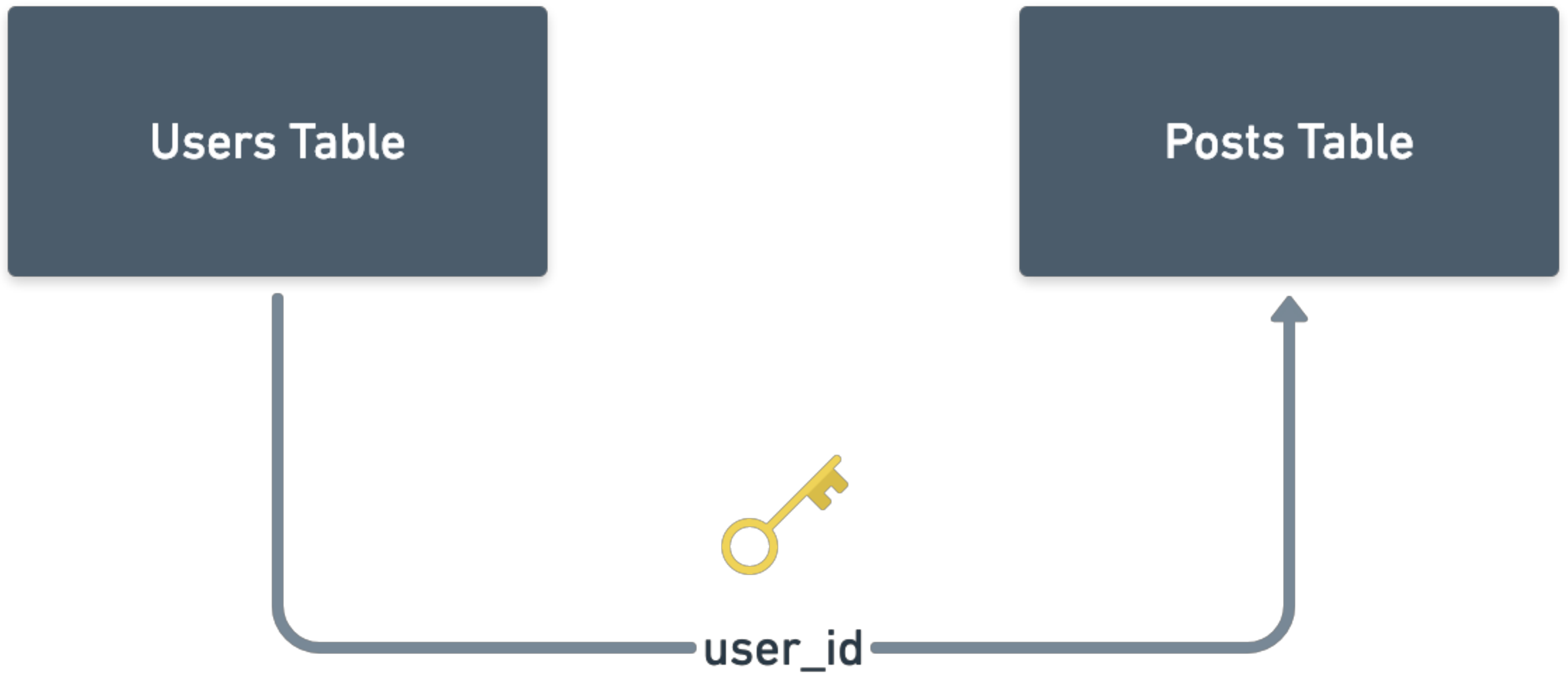
- Foreign keys
- Joins
- Updating data
- Deleting data

Foreign keys

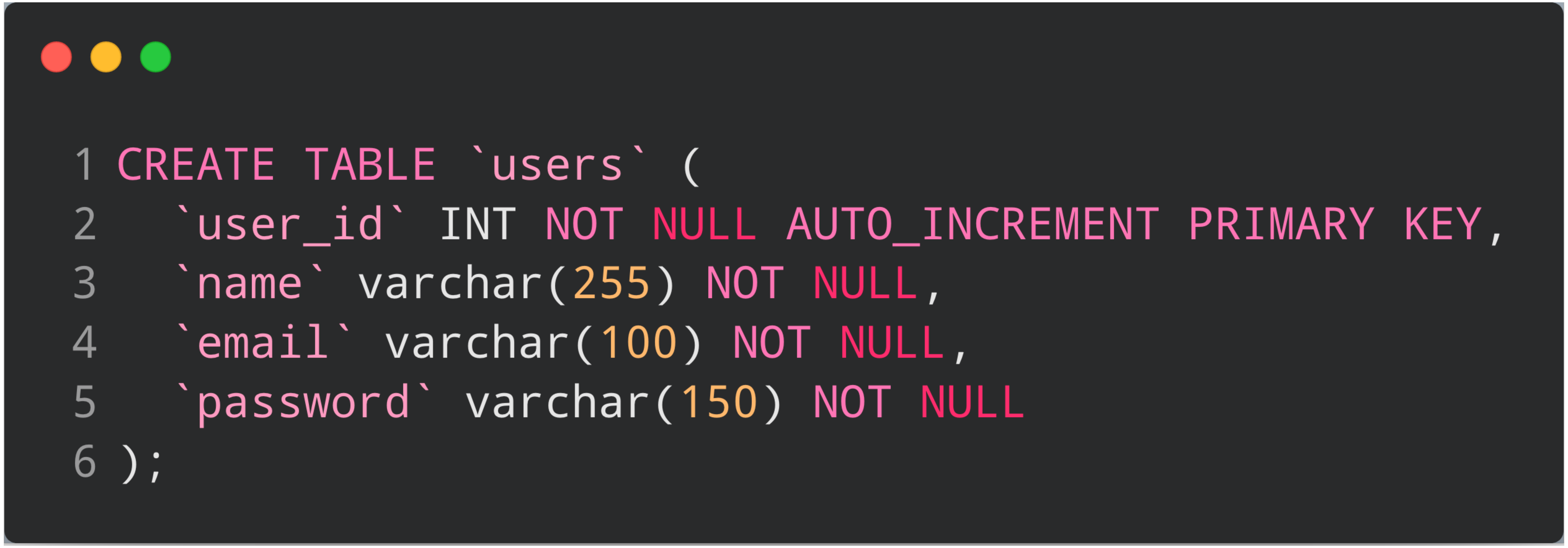
- A foreign key **joins** two or more tables together
- This **join** is usually on a **primary key**
- Foreign keys are handy for maintaining a formal, solid relationship between tables
- They're handy for maintaining the health of your data

Creating a foreign key

We’ll create this simple relationship

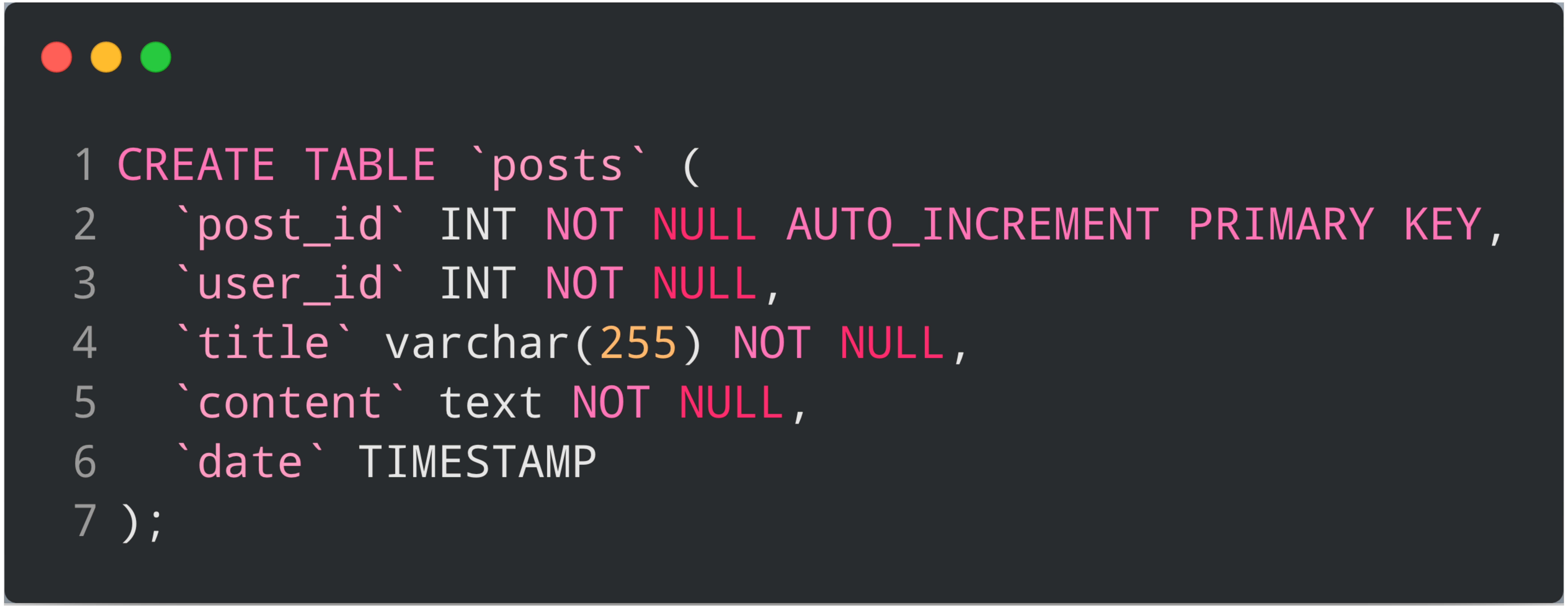


Creating a foreign key: step 1



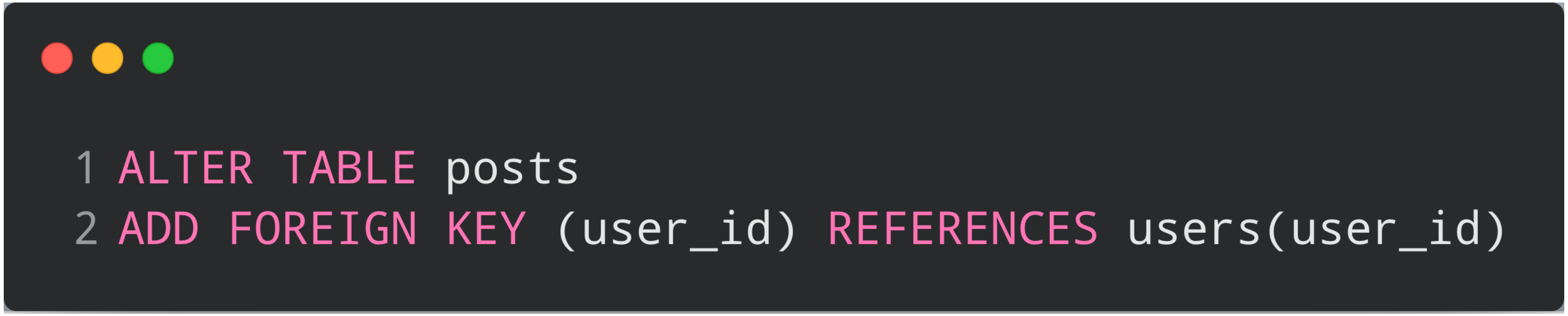
```sql
CREATE TABLE `users` (
  `user_id` INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  `name` varchar(255) NOT NULL,
  `email` varchar(100) NOT NULL,
  `password` varchar(150) NOT NULL
);
```

Creating a foreign key: step 2



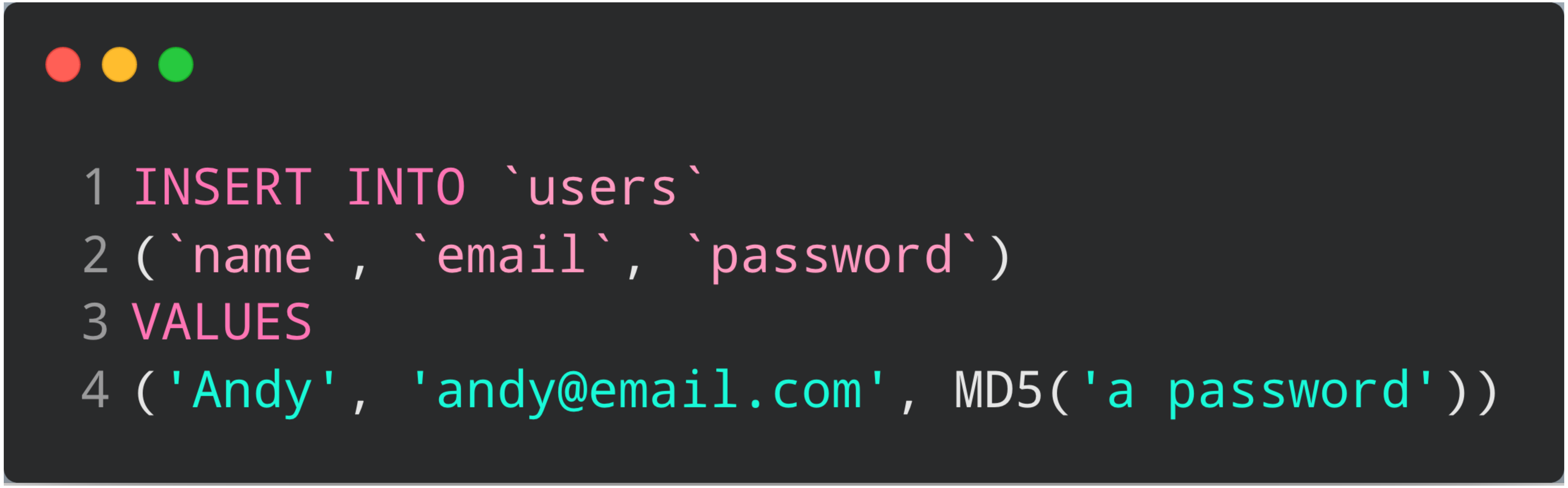
```sql
CREATE TABLE `posts` (
  `post_id` INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  `user_id` INT NOT NULL,
  `title` varchar(255) NOT NULL,
  `content` text NOT NULL,
  `date` TIMESTAMP
);
```

Creating a foreign key: step 2


```sql
ALTER TABLE posts
ADD FOREIGN KEY (user_id) REFERENCES users(user_id)
```

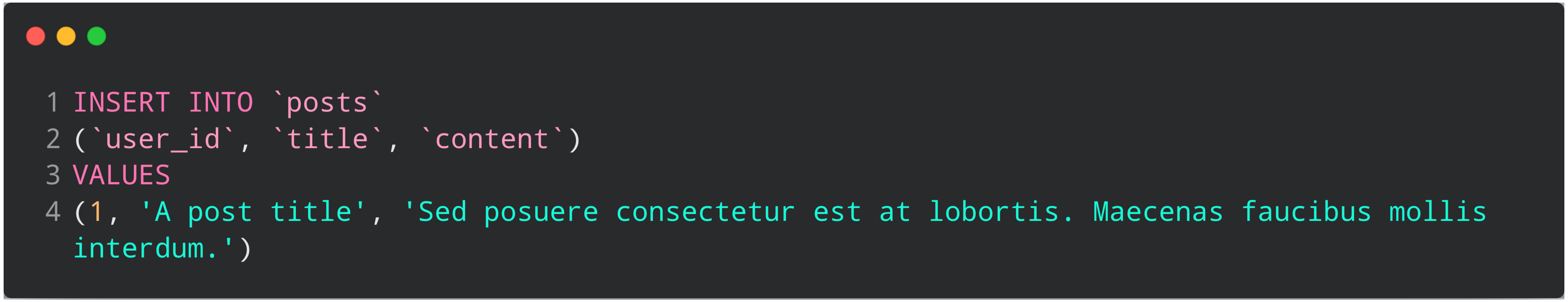
Now we can a user and a post that have a proper link

Creating a foreign key: step 3



```sql
INSERT INTO `users`
(`name`, `email`, `password`)
VALUES
('Andy', 'andy@email.com', MD5('a password'))
```

Creating a foreign key: step 4

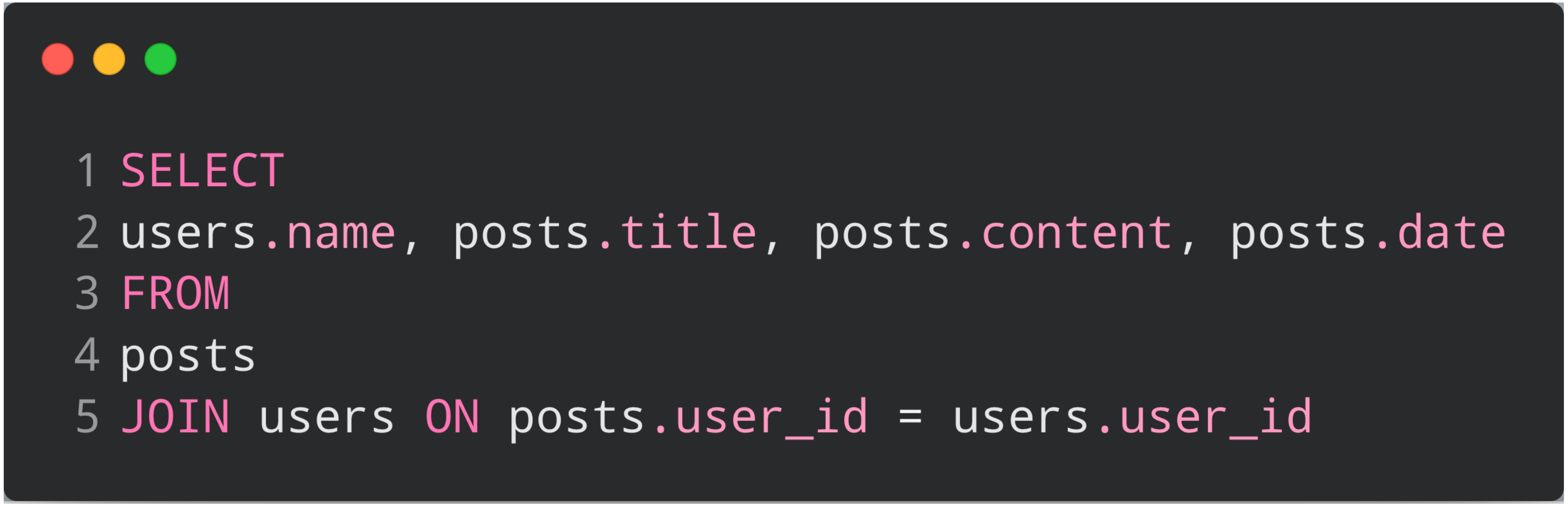


```sql
INSERT INTO `posts`
(`user_id`, `title`, `content`)
VALUES
(1, 'A post title', 'Sed posuere consectetur est at lobortis. Maecenas faucibus mollis
interdum.')
```

What’s the use of this?

One thing: we can write efficient queries!

A JOIN query



```sql
SELECT
users.name, posts.title, posts.content, posts.date
FROM
posts
JOIN users ON posts.user_id = users.user_id
```

Server: localhost:3306 » Database: abell_sandbox

Structure | SQL | Search | Query | Export | Import | Operations | Routines | More

Recent | Favorites

- abell_ct4009
- abell_sandbox
  - New
  - posts
  - users
- information_schema

Show query box

Showing rows 0 - 0 (1 total, Query took 0.0002 seconds.)

```
SELECT users.name, posts.title, posts.content, posts.date FROM posts JOIN users ON posts.user_id = users.user_id
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

+ Options

name	title	content	date
Andy	A post title	Sed posuere consectetur est at lobortis. Maecenas ...	2019-01-28 11:12:06

Show all | Number of rows: 25 Filter rows: Search this table

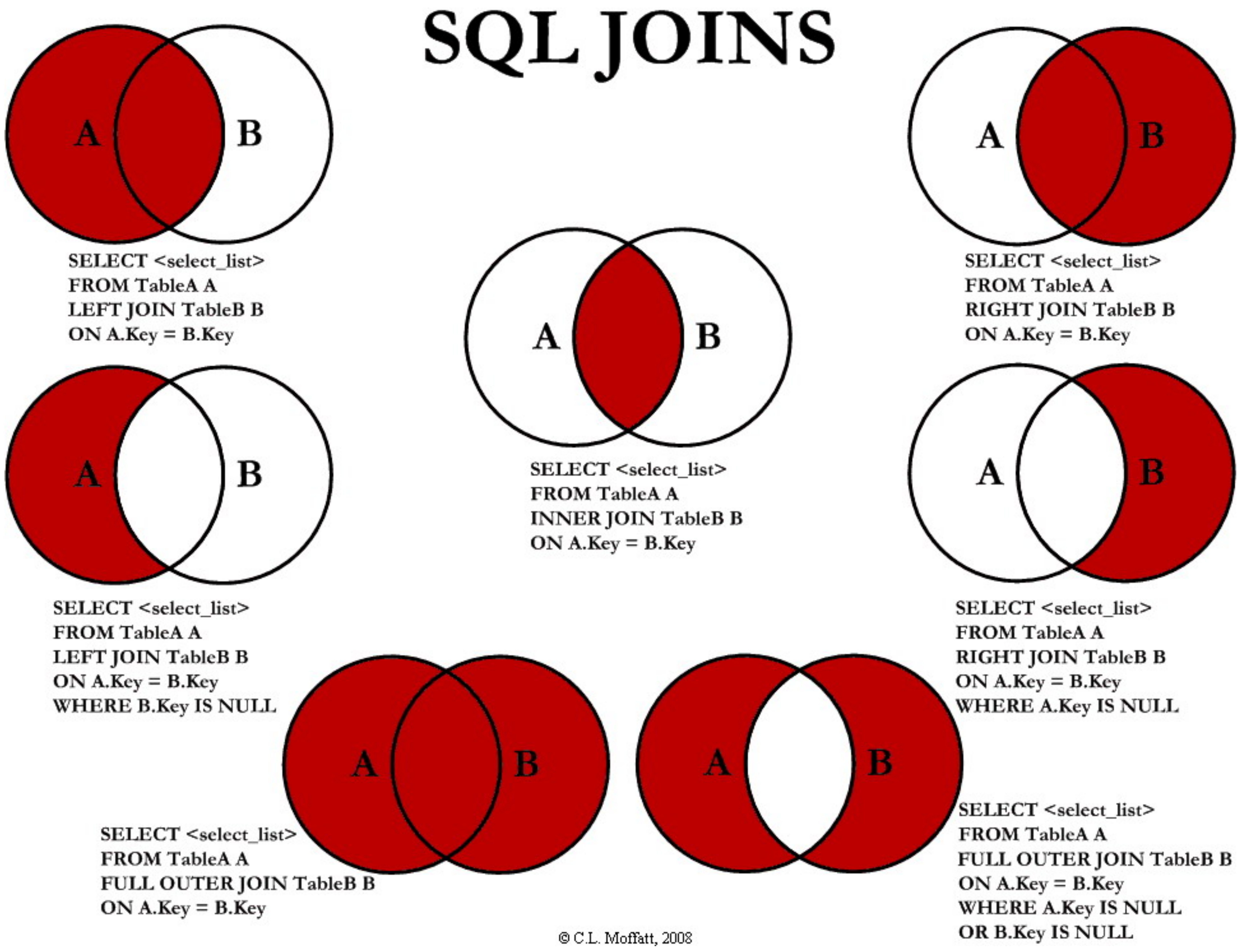
Query results operations

Print | Copy to clipboard | Export | Display chart | Create view

Console

Query efficiency

- Because we are **joining** our tables together at the point where we **select** the data, we can pull from multiple tables in one query
- This is **paramount** if your MySQL database lives on a different server to your application
- Queries are much faster when there's lots of data spread out
- You keep the relevant data in the relevant tables which means you have **single sources of truth**



SQL JOINS

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```

```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

© C.L. Moffatt, 2008

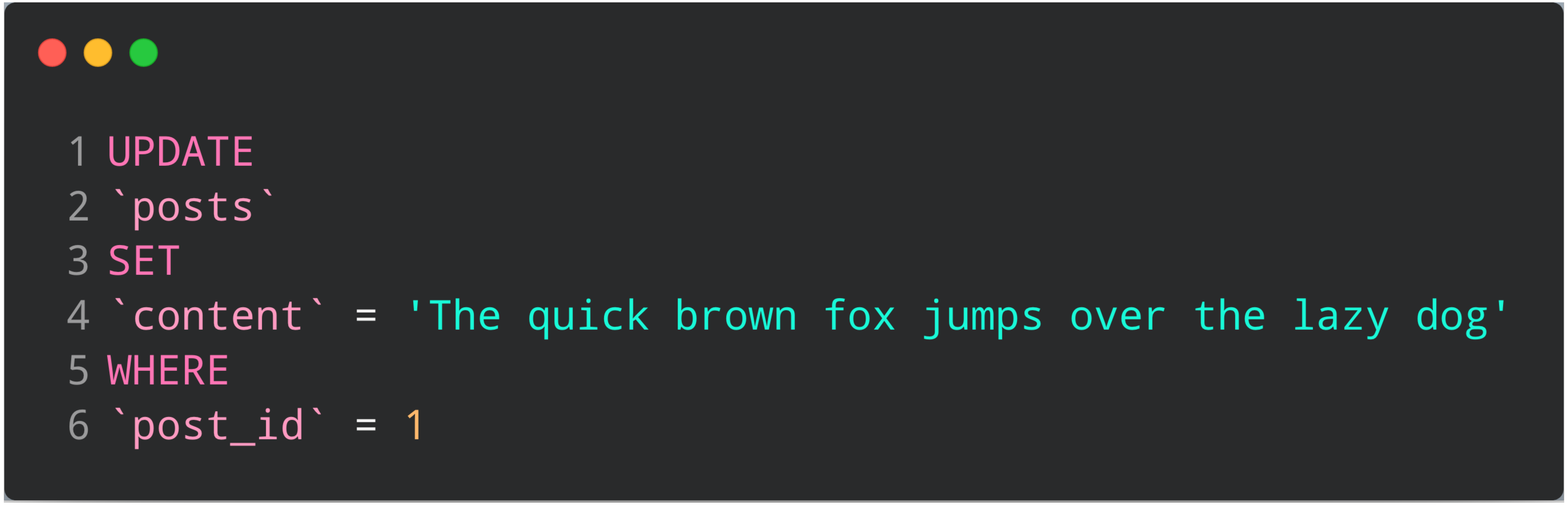
https://www.codeproject.com/Articles/33052/Visual-Representation-of-SQL-Joins

Updating data

To **UPDATE** a record, you must first know what it’s **id** is.

Let’s use the example of the post that we just created and change the copy from placeholder text, to something more readable.

Update command

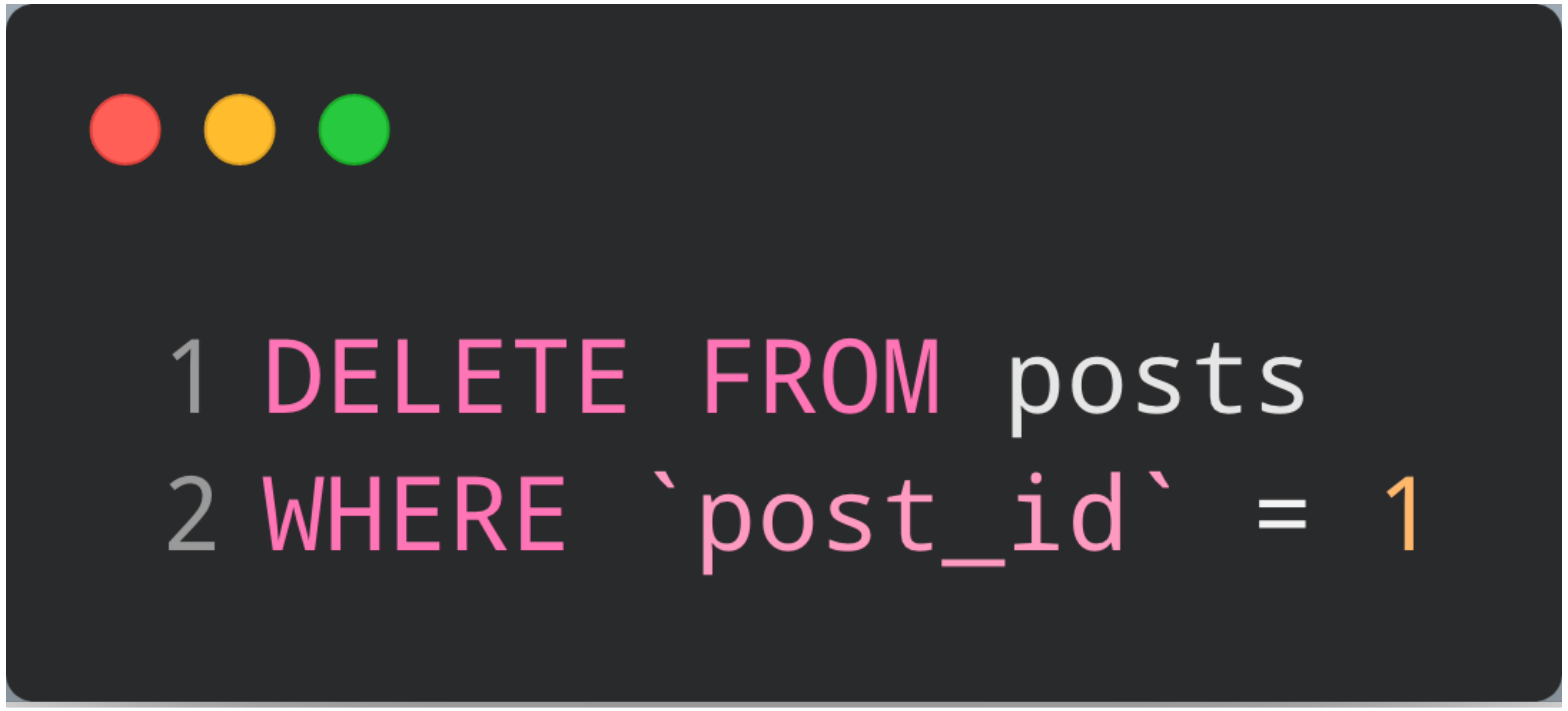


```sql
UPDATE
`posts`
SET
`content` = 'The quick brown fox jumps over the lazy dog'
WHERE
`post_id` = 1
```

BEWARE ⚠️

If you don’t specify a **WHERE CLAUSE,** every single row in your table will be updated with the new data!!!

Deleting data



```sql
DELETE FROM posts
WHERE `post_id` = 1
```

BEWARE ⚠️

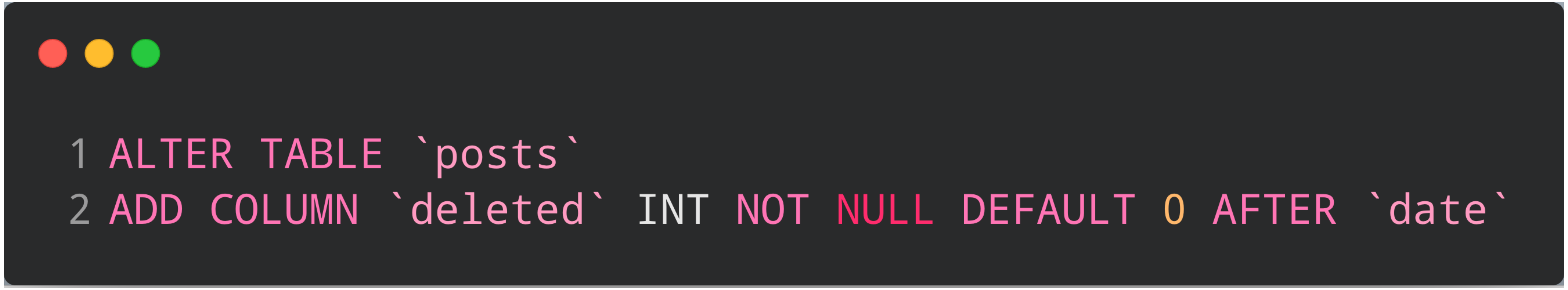
If you don’t specify a **WHERE CLAUSE,** every single row in your table will be **DELETED**!!!

Soft deletes

A **soft delete** is where you use a flag to determine deleted status, rather than deleting the record.

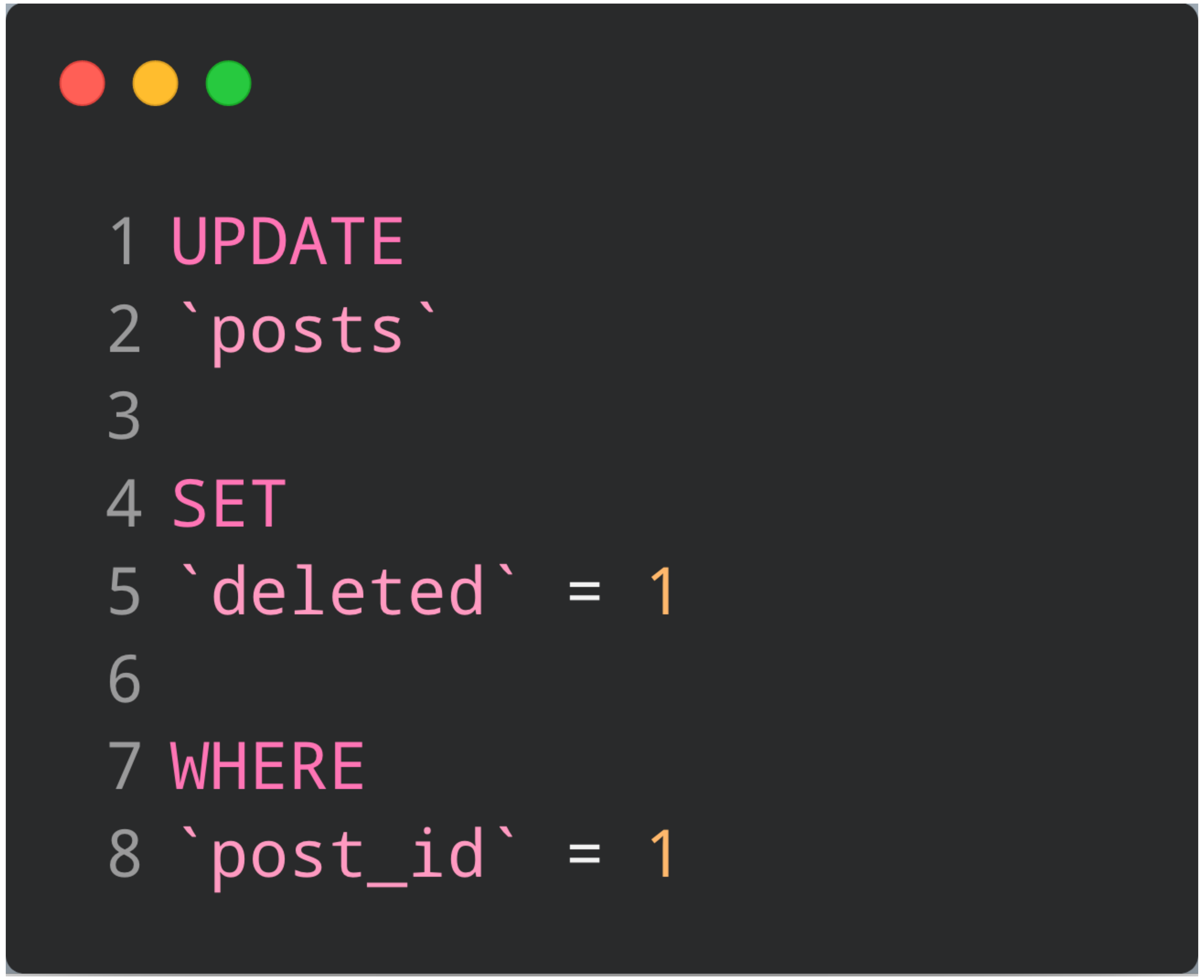
This is a good way of keeping hold of data and preventing accidents.

Altering our post table



```sql
ALTER TABLE `posts`
ADD COLUMN `deleted` INT NOT NULL DEFAULT 0 AFTER `date`
```

New delete command

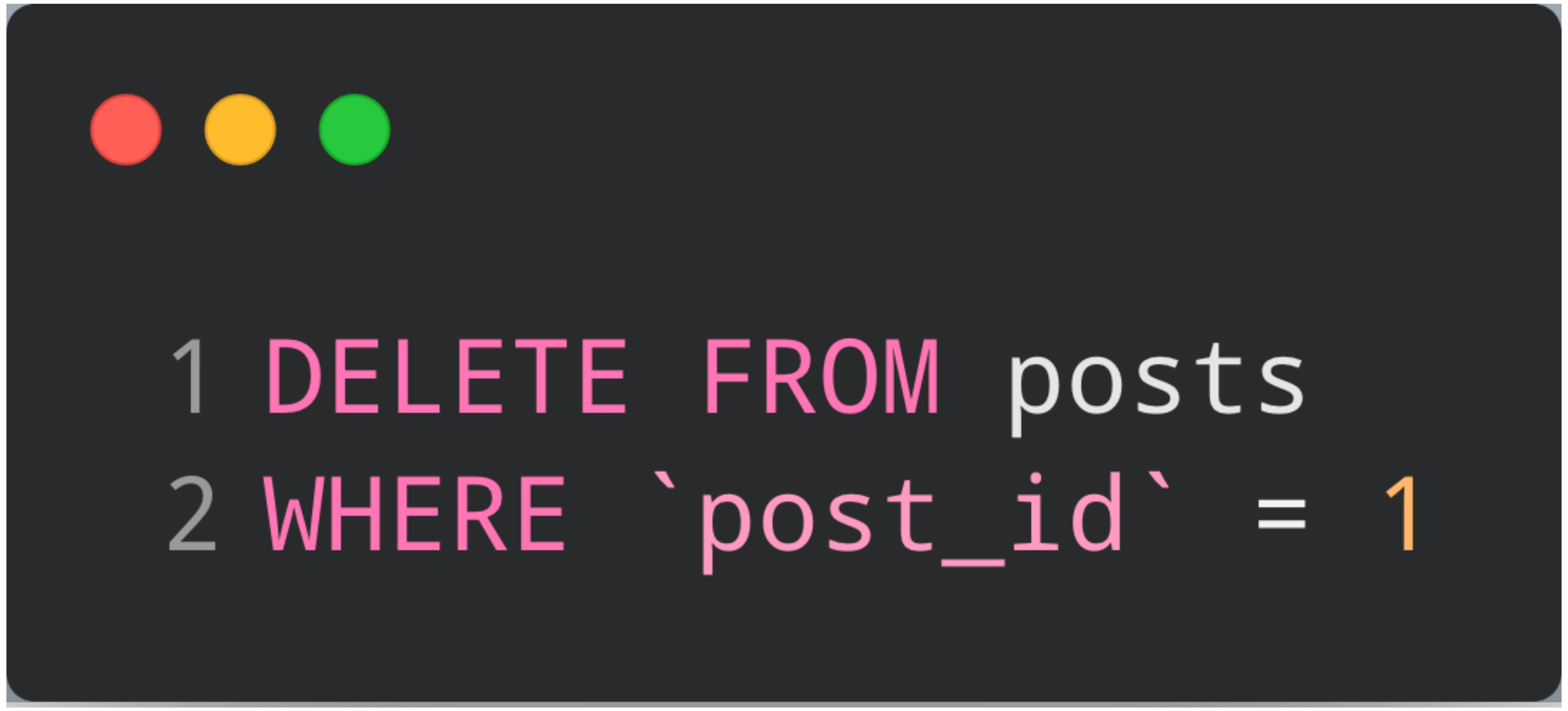


```sql
UPDATE
`posts`

SET
`deleted` = 1

WHERE
`post_id` = 1
```

Old delete command



```sql
DELETE FROM posts
WHERE `post_id` = 1
```

Recap

- We learned about foreign keys
- We learned about joins
- We learned about updating data
- We learned about deleting data (and soft deletes)