About this Demo

This is a PDF demo of the book, Wizard's SQL Recipes for WordPress. It contains the entire Table of Contents and part of the third chapter. The complete book contains over 200 pages, packed with 300+ SQL recipes.

Note! This demo does not contain all pages from the book. So any links pointing to missing pages will not work. Of course, all links work perfectly in the complete version of the book :)

WIZARD'S COLLECTION

SQL RECIPES

WORDPRESS COOKBOOK

DELUXE CODE SNIPPETS

300+ MAGICAL RECIPES

OPTIMIZE PERFORMANCE

LEVEL UP YOUR DATABASE SKILLS

Wizard's Collection: SQL Recipes for WordPress

by

Jeff Starr

CONTENTS

1 WELCOME

2 SQL OVERVIEW

3 WP POSTS & PAGES

WP POST META

5 WP COMMENTS

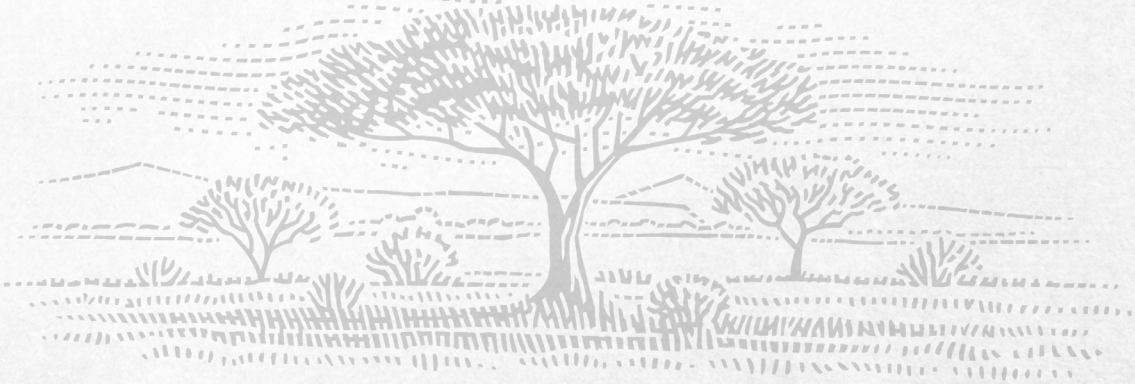
WP USERS

WP TAXONOMIES

8 WP OPTIONS

9 MORE RECIPES

10 OPTIMIZE WP



WP POSTS & PAGES

Welcome to the posts table

Hands down, the #1 most common thing that people do with SQL and the WordPress database is make changes to post and page content. So we're putting the post and page recipes right up front. This chapter brings together all of the best, most commonly used SQL recipes for working with the WordPress `posts` table.

The `posts` table, or `wp_posts` if using the default prefix, is where post content is stored. This includes content for posts, pages, and all other post types. The `posts` table contains over 20 columns, including `post_author`, `post_date`, `post_status`, `post_title`, and `post_type`. It's generally the largest table (size-wise) in the WordPress database.

Find any string in post content

Let's kick things off with a common and simple recipe: how to find all posts that include a certain string in the post content. If the string is "lorem", we enter this query:

```
SELECT * FROM wp_posts WHERE post_content LIKE '%lorem%';
```

That returns all posts that include the word "lorem" anywhere in the `post_content` field of

the posts table. Because post content usually includes more than one word, we are using the wildcard operator `%` to tell SQL that we want to match "lorem" *anywhere* it appears in the post content. To instead match any post that includes only a *specific* word or phrase, simply omit the `%` like this:

```
SELECT * FROM wp_posts WHERE post_content LIKE 'lorem ipsum';
```

That query returns all posts where the `post_content` field includes only "lorem ipsum" and nothing else. Remember to replace the `wp_` prefix on `wp_posts` if needed.

Find all password-protected posts

In the previous recipe, we query the `post_content` column. Here is a similar recipe where we query the `post_password` column to find all posts that are password-protected.

```
SELECT * FROM wp_posts WHERE post_password != '';
```

The above query can be modified to check for non-empty values for any column in the `wp_posts` table. Simply replace `post_password` with the column that you want to search.

Replace any string in post content

This recipe enables us to find and replace any string in post content. For example, we can replace all instances of the word "apprentice" with the word "wizard". Before making changes, let's do a simple `SELECT` query to get an idea of how many rows (records) are involved. So we enter the following `SELECT` query:

```
SELECT * FROM wp_posts WHERE post_content LIKE '%apprentice%';
```

The above query returns all posts that include the word “apprentice”. Once ready to make changes, we can find and replace all instances with a simple `UPDATE` query:

```
UPDATE wp_posts SET post_content = REPLACE(post_content, 'apprentice',
'wizard');
```

That query will replace all instances of “apprentice” with “wizard” in all post content. Of course, you can edit the query to search and replace any word or phrase that is required.

Tip:

The `REPLACE()` function replaces all occurrences of one string with another. No wildcard characters are required. The `REPLACE()` function operates in wildcard fashion.

Replace old URL with new URL

Another recipe that I have used countless times in my WordPress adventures. The following query replaces an old URL with a new one:

```
UPDATE wp_posts SET post_content = REPLACE(post_content, 'https://old-url.
com', 'https://new-url.com');
```

Remember to change the URLs to match your own.

Update URLs from HTTP to HTTPS

Many sites are making the switch from `http://` (non-SSL/TLS) to `https://` (SSL/TLS). As a part of the process, it is important to update the URLs of any links and images in post

content, comments, and elsewhere. Doing this manually can require much time and work. Instead, running a quick SQL query can do the job quickly and easily. Here is a query to replace all instances of `http://` with `https://` for any URL found in the posts table.

```
UPDATE wp_posts SET post_content = REPLACE(post_content, 'http://example.
com', 'https://example.com');
```

This same general query can be modified to replace URLs in other tables. Simply change the table name and *both* instances of the column name. For example, the following query updates all URLs in the `wp_comments` table:

```
UPDATE wp_comments SET comment_content = REPLACE(comment_content, 'http://
example.com', 'https://example.com');
```

There are many uses for the previous `UPDATE` recipe. By changing the table name and both instances of the column name, `comment_content` in this example, this recipe can be used to find and replace any string in any WordPress table.

Update URLs for media only

Using the same basic formula as above, we can change URLs found only in media such as images, audio, and video by including a bit of context. The trick is to include the `src` attribute in the `REPLACE()` function, for example:

```
UPDATE wp_posts SET post_content = REPLACE(post_content, 'src="http://old-
url.com', 'src="https://new-url.com');
```

This works because only images and other media use the `src` attribute for URLs. Here is a

list of all the HTML elements that use the `src` attribute:

- `<audio>`
- `<embed>`
- `<iframe>`
- `<img>`
- `<input>`
- `<script>`
- `<source>`
- `<track>`
- `<video>`

The previous recipe will affect any of the above tags that match the specified string. Normally the `<script>` tag is not allowed in WordPress post content, but it is possible for plugins to include it. This is why it's wise to do a `SELECT` query before updating anything, so you can verify potential changes before they happen, and adjust the query accordingly.

Update URLs for links only

Similar to the previous recipe, here we update URLs that appear only in hyperlinks. Instead of including the `src` attribute, we include the `href` attribute, for example:

```
UPDATE wp_posts SET post_content = REPLACE(post_content, 'href="http://old-url.com', 'href="https://new-url.com');
```

In addition to hyperlinks, this query affects any `<area>` tag that includes an `href` attribute and matching URL. In general, the `href` attribute may be used with any of these tags:

- `<a>`
- `<area>`
- `<base>`
- `<link>`

Of these, the `<base>` and `<link>` tags should not be found in post content. Also note that the `<area>` tag may be used in conjunction with the `<map>` tag. So just be aware of these nuances and adjust your queries accordingly.

Replace weird characters in post content

Especially for databases that have been transferred from one server to another, it is fairly common to find "weird" encoded characters in post content, comments, and elsewhere. This can happen due to encoding differences when working with databases (not just WordPress). The following SQL recipes can help to clean up any weird characters by replacing them with their unencoded equivalents.

```
UPDATE wp_posts SET post_content = REPLACE (post_content, 'â€œ', '“');
UPDATE wp_posts SET post_content = REPLACE (post_content, 'â€ ', '”');
UPDATE wp_posts SET post_content = REPLACE (post_content, 'â€™', '’');
UPDATE wp_posts SET post_content = REPLACE (post_content, 'â€˜', '‘');
UPDATE wp_posts SET post_content = REPLACE (post_content, 'â€”', '–');
UPDATE wp_posts SET post_content = REPLACE (post_content, 'â€“', '—');
UPDATE wp_posts SET post_content = REPLACE (post_content, 'â€¢', '-');
UPDATE wp_posts SET post_content = REPLACE (post_content, 'â€¦', '…');
```

You can enter each of the above queries one at a time, or enter them all at once. This is why the recipes in this book end each query with a semicolon `;`. Without it, each query would need entered individually. When the semicolon is included, the queries can be run together. To go further with the above recipe, visit the original tutorial at DigWP.com.

Delete any string in post content

Using the same basic query that we use for replacing strings, we also can *delete* any unwanted strings. For example, if we want to delete all instances of an old URL from post content, we can enter the following query:

```
UPDATE wp_posts SET post_content = REPLACE(post_content, 'http://old-url.
com', '');
```

Notice the trick here: the third parameter is left empty `''` in the `REPLACE()` function. Basically, this query says, "replace all instances of `http://old-url.com` with nothing", which effectively is the same as deletion.

Tip:

For further information on the previous recipes, check out Remove Content from the WordPress Database, and MySQL Magic: Find and Replace Data.

Remove unwanted shortcodes from post content

Here is a quick query to delete all instances of a shortcode in post content. Most plugins do not remove their shortcodes from the database. So when the plugin is removed, their old shortcodes will appear literally as `[example]` in your post content, which is confusing to visitors and just looks bad. So to remove an unwanted shortcode, run this query:

```
UPDATE wp_posts SET post_content = REPLACE(post_content, '[example]', '');
```

Replace `[example]` with the name of the actual shortcode that you want to delete. If you have very many posts, you can use the following alternate query instead. It is more specific and thus a bit faster while producing the same results.

```
UPDATE wp_posts SET post_content = REPLACE(post_content, '[example]', '')
WHERE post_content LIKE '%\\[example\\]%';
```

Tip:

In the previous recipe, what does `%\\[` mean? The first character is a wildcard `%`. Then two slashes `\\` escape the square bracket. Two slashes are required. The first slash escapes the second slash, which in turn escapes the bracket character.

Find all posts with a specific post status

Here are some quick recipes for showing all posts with a specific post status. This is useful for viewing all posts that are in draft status, published status, or any post status.

```
# get all draft posts
SELECT * FROM wp_posts WHERE post_status = 'draft';

# get all published posts
SELECT * FROM wp_posts WHERE post_status = 'publish';

# get all private posts
SELECT * FROM wp_posts WHERE post_status = 'private';
```

We can replace the value of `post_status` as needed to find posts that have other statuses. By default, WordPress makes use of the following post statuses: `publish`, `future`, `draft`, `pending`, `private`, `trash`, `auto-draft`, and `inherit`. For further information about post statuses, custom statuses and more, visit the Post Status documentation at WordPress.org.

Tip:

Wondering about the use of # in some of the SQL recipes? As with PHP and other scripting languages, SQL allows for inline comments in SQL queries. Comments are for informational purposes only and ignored by SQL. Learn more about SQL comments.

Delete all posts with a specific post status

Just as SQL makes it easy to *select* all posts with a given Post Status, it also makes it easy to *delete* them. For example, to delete all posts with either `draft` or `auto-draft` status, enter both of the following queries. You can enter the queries together or individually.

```
# delete all draft posts
DELETE FROM wp_posts WHERE post_status = 'draft';

# delete all auto-draft posts
DELETE FROM wp_posts WHERE post_status = 'auto-draft';
```

The above queries delete all matching posts, but they do not delete any associated metadata. To delete all matching posts **and** their related metadata, enter these queries instead:

```
# delete all draft posts and metadata
DELETE a,b,c FROM wp_posts a
LEFT JOIN wp_term_relationships b ON (a.ID = b.object_id)
LEFT JOIN wp_postmeta c ON (a.ID = c.post_id)
WHERE a.post_status = 'draft';

# delete all auto-draft posts and metadata
DELETE a,b,c FROM wp_posts a
LEFT JOIN wp_term_relationships b ON (a.ID = b.object_id)
LEFT JOIN wp_postmeta c ON (a.ID = c.post_id)
WHERE a.post_status = 'auto-draft';
```

The above recipes delete all draft posts: drafts and auto-drafts. To instead delete posts and metadata for some other post status, like inherit, pending, or trash, replace this line:

```
WHERE a.post_status = 'draft';
```

…with this line:

```
WHERE a.post_status = 'post-status';
```

…and then replace `post-status` with the name (a.k.a., slug) of your post status.

Tip:

The draft status is for posts that are created but not yet published. The auto-draft status is for the copies of posts that WordPress makes automatically as the post is edited. Over time, the number of drafts and auto-drafts can really add up.

Find and delete all posts with a specific post type

By default, WordPress stores a copy of each saved draft or published change for each post. So for example, if a post has been published and edited several times, it will have several revisions stored in the database. As one might suspect, the post type for post revisions is `revision`. Over time, revisions can consume a LOT of space. To clean things up, here are a couple of queries to first select and then delete all post revisions.

```
# select all post revisions
SELECT * FROM wp_posts WHERE post_type = 'revision';

# delete all post revisions
DELETE FROM wp_posts WHERE post_type = 'revision';
```

To find and delete some other post type (e.g., `post`, `page`, `attachment`, `nav_menu_item`), we can replace both instances of `revision` with the name of the post type. Note that this technique deletes the posts only. To delete all revisions **and** any related metadata, check out the next recipe. Also, for a complete list of default post types and more, visit the Post Type documentation at WordPress.org.

Delete post type and related metadata

Whereas the previous recipe deletes only revision posts, the following recipe deletes revision posts **and** all associated metadata. So this recipe is a more *complete* removal of posts with a specific post status. Note that these recipes work for any post type. Simply replace `revision` with the status that you want to target.

```
DELETE a,b,c FROM wp_posts a
LEFT JOIN wp_term_relationships b ON (a.ID = b.object_id)
LEFT JOIN wp_postmeta c ON (a.ID = c.post_id)
WHERE a.post_type = 'revision';
```

Note that this query alternately may be written as three individual queries, for example:

```
DELETE FROM wp_posts WHERE post_type = 'revision';

DELETE FROM wp_postmeta WHERE post_id NOT IN (SELECT ID FROM wp_posts);

DELETE FROM wp_term_relationships WHERE object_id NOT IN (SELECT ID FROM
wp_posts);
```

Either way works equally well. To learn more about how the `JOIN` clause works, check out A step-by-step walkthrough of SQL Inner Join over at SQLShack.com.

Tip:

To stop WordPress from creating any post revisions, add the following line to your site's `wp-config.php` file. Include it just before the line that says, "That's all, stop editing [...]".

```
define('WP_POST_REVISIONS', false);
```

For more information, visit the wp-config documentation at WordPress.org.

Find & delete posts with empty excerpts

This recipe returns all posts that have an empty excerpt field. This is useful when you want to make sure that all posts have an excerpt.

```
SELECT * FROM wp_posts WHERE post_excerpt IS NULL OR post_excerpt = '';
```

To delete all posts that have empty excerpts, run this query:

```
DELETE FROM wp_posts WHERE post_excerpt IS NULL OR post_excerpt = '';
```

Find & delete posts with non-empty excerpts

Similar to the previous recipe, this query returns all posts that have a non-empty excerpt.

```
SELECT * FROM wp_posts
WHERE post_excerpt IS NOT NULL AND post_excerpt <> '';
```

To delete all posts that have non-empty excerpts, run this query:

```
DELETE FROM wp_posts WHERE post_excerpt IS NOT NULL AND post_excerpt <> '';
```

Tip:

Reminder: <> means “not equal to”. Visit chapter 2 for more information about syntax.

Find & delete posts with empty content

Another "find empty" recipe, this query returns all posts that have an empty content field.

```
SELECT * FROM wp_posts WHERE post_content IS NULL OR post_content = '';
```

To delete all posts that have empty content, run the following query:

```
DELETE FROM wp_posts WHERE post_content IS NULL OR post_content = '';
```

Find & delete posts with non-empty content

This recipe returns all posts that have non-empty post content.

```
SELECT * FROM wp_posts WHERE post_content IS NULL OR post_content <> '';
```

Remember, it's always possible to tweak SQL recipes to query other tables and columns. For example, to find all non-empty posts of a certain post type, we can do this:

```
SELECT * FROM wp_posts WHERE post_content IS NULL OR post_content <> '' AND
post_type = '{post_type}';
```

This is the same query as before, only here we have added an `AND` clause that specifies the post type. Here we are specify `{post_type}` as the post type. You can change that to match whatever post type you want to target.

For the sake of completeness, here is a query to delete all posts that have non-empty content. You don't actually want to run this query on a live production site, as it effectively deletes any post that is not empty. So just for reference:

```
DELETE FROM wp_posts WHERE post_content IS NULL OR post_content <> '';
```

Change post author

A common task in WordPress is to change the author on specific posts. Fortunately, SQL can chop hours of work down to seconds. To do it, we first need to get the correct user IDs. You can use the following query to get a list of all users and their associated IDs:

```
SELECT ID, user_login, display_name FROM wp_users;
```

That query will return something like this:

```
 ID    |    user_login         |    display_name
-------+-----------------------+-------------------
 1     |    wordpress_user     |    WordPress User
 2     |    another_user       |    Another User
 3     |    and_another        |    And Another
```

Notice the ID column. Use that to locate the ID of both the original/current post author and the new post author. Once you have the required user IDs, here is a recipe to change the author of all matching posts. For example, if our author IDs are 1 and 2, we do this:

```
UPDATE wp_posts SET post_author = 1 WHERE post_author = 2;
```

Keep going!

Get over 300 SQL recipes for WordPress:

https://mon.co/wizards-sql

ABOUT THE AUTHOR

Jeff Starr is a web developer and book author with over 15 years of experience. Jeff's work focuses on WordPress, web security, and helping people to succeed online. He sells premium WordPress plugins at Plugin Planet and runs his own web development business at Monzilla Media. Jeff co-authors the book and blog, Digging into WordPress, and is the author of The Tao of WordPress, WordPress Themes In Depth, and .htaccess made easy. Jeff also shares tutorials and snippets at Perishable Press and WP-Mix.com.

Follow Jeff Starr on Twitter

Connect with Jeff Starr on Facebook

Jeff's WordPress Developer Profile