WooCommerce Database Walkthrough

Rodolfo Melogli

Business Bloomer

Context

Why database knowledge changes everything

Why understanding the database matters

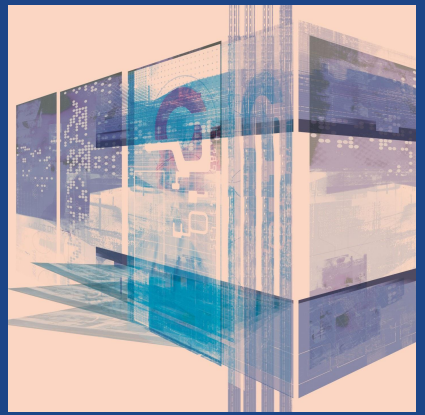
1. **Everything in WooCommerce lives in tables**
   Orders, products, customers—all are rows of data.
2. **Custom code relies on accurate data access**
   Understanding structure prevents wrong queries or logic errors.
3. **Performance begins with smart queries**
   Poorly written SQL or PHP slows stores dramatically.

Common Pain Points

1. **Guessing where data is stored**
   Many developers still look in postmeta first.
2. **Inefficient or broken custom reports**
   Reporting fails without knowing the schema.
3. **Slow or unoptimized queries**
   SELECT * FROM chaos—avoid that with knowledge.

WooCommerce's Architecture in Context

1. **Built on WordPress, not standalone**
   Uses core tables (posts, users, meta) extensively.
2. **Recent move to custom tables (HPOS)**
   Transition designed for performance and clarity.
3. **Plugins add even more tables**
   Subscriptions, Payments, Analytics—all extend schema.



What You'll Learn Today

1. **Table-by-table schema breakdown**
   You'll see exactly how data connects internally.
2. **Real examples**
   Not just theory—actual query use cases.
3. **Actionable knowledge for developers**
   Walk away able to query and debug confidently.

Rodolfo Melogli - Business Bloomer

Business Bloomer is the **network** that helps WooCommerce freelancers, developers, and makers grow their skills, revenue, and confidence. Community, videos, templates, resources, and real-time WooCommerce support—all in one professional hub.

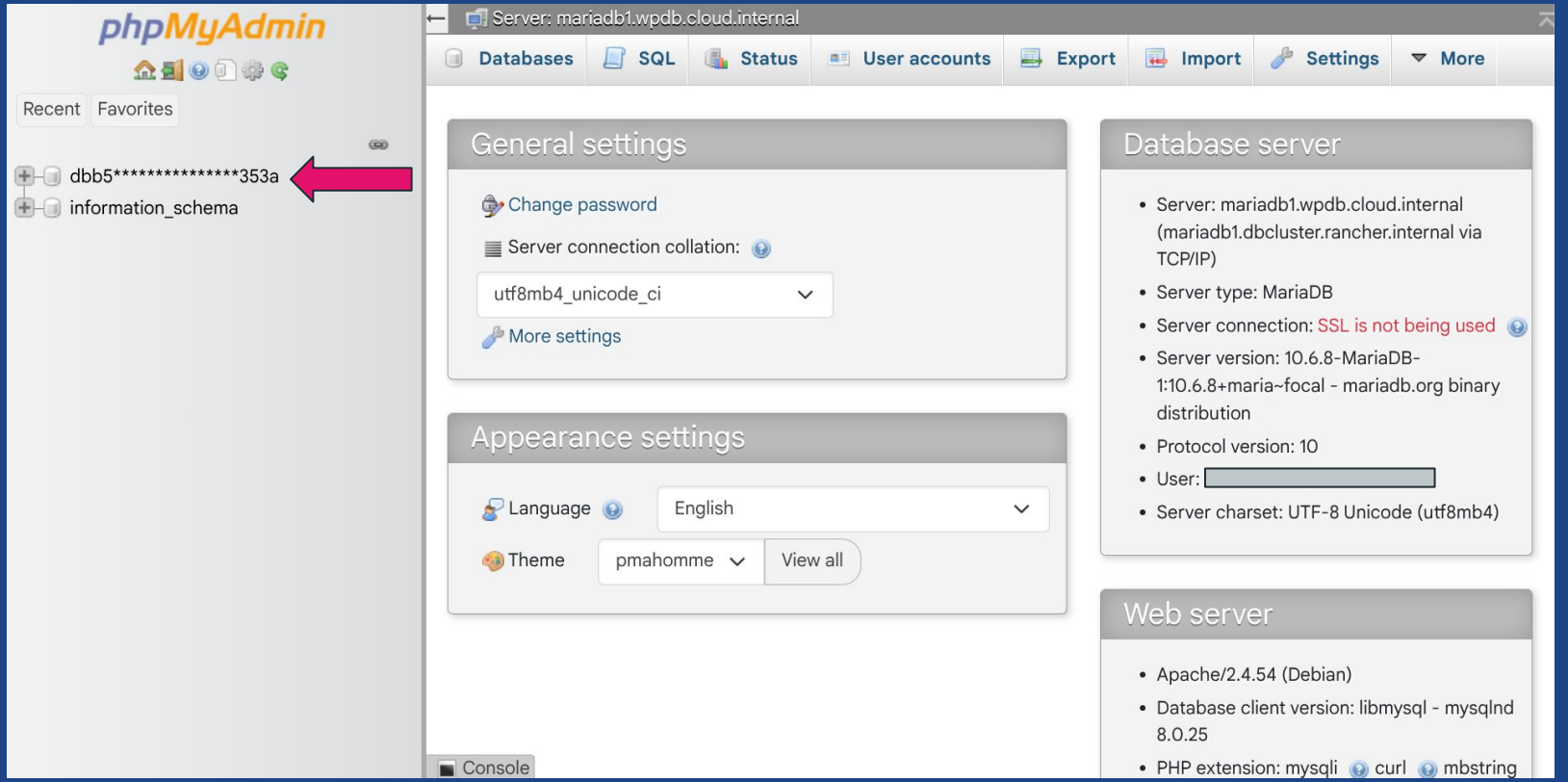
Get Your Hands Dirty

Time to open phpMyAdmin (or your favorite DB tool) and explore the WP database

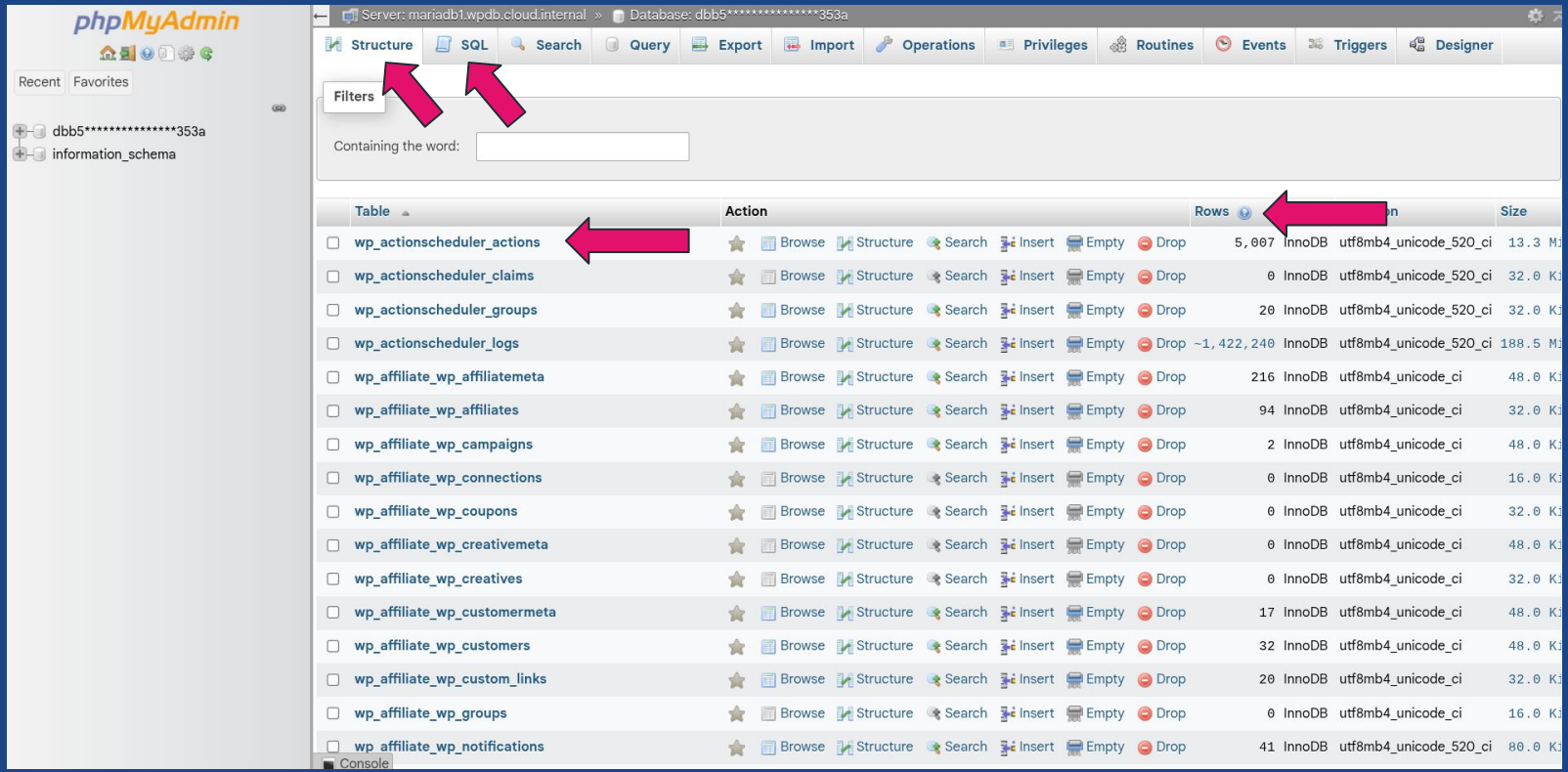
Hosting Control Panel

The phpMyAdmin name is a mixture of **PHP** as the language it uses, **MySQL** as the database it manages and **administration** as the activity it handles.

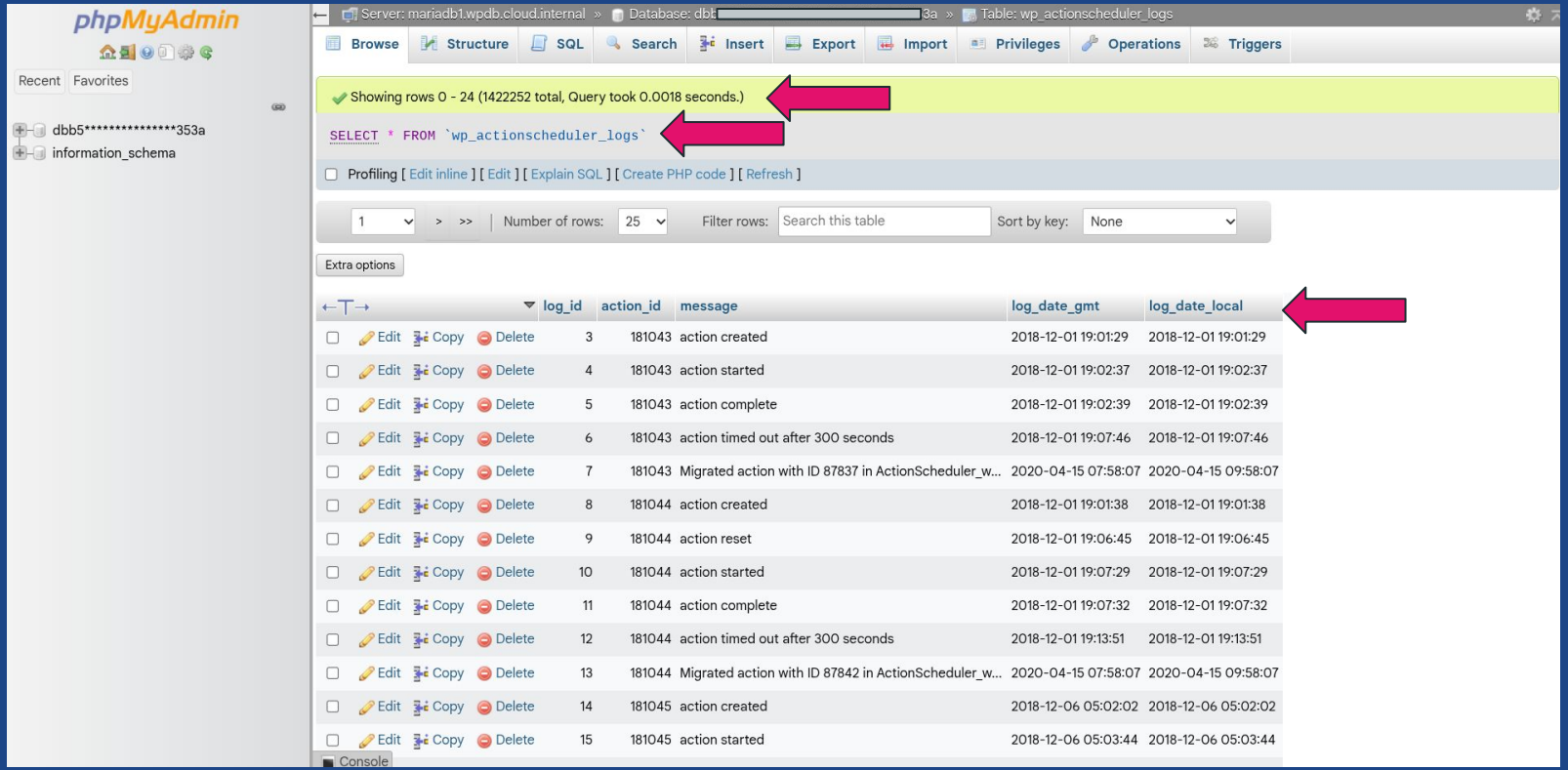
phpMyAdmin - Landing Page



phpMyAdmin - Database Tables



phpMyAdmin - Database Table

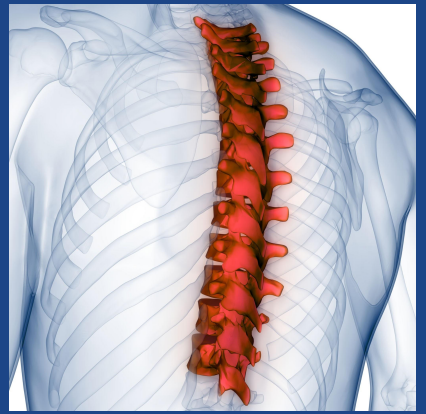


WP Tables

Understanding WooCommerce's WordPress foundation

WordPress as WooCommerce's Backbone

1. WooCommerce extends, not replaces, WordPress
2. WP uses the DB to store, retrieve, and display all the content. This includes posts, pages, comments, and more
3. WordPress uses a database management system called MySQL



wp_posts and wp_postmeta

1. **Posts, pages, or any other custom post type**
2. **Each row in the wp_posts table represents a single post**
3. **The wp_postmeta table allows you to store additional information about each post**
   The post meta are also often referred to as custom fields.

wp_comments and wp_commentmeta

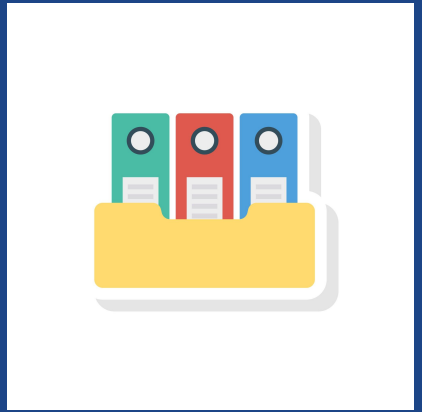
1. Whenever someone comments on a post or page, this table is where that comment is stored
2. Each row in the wp_comments table represents a single comment
3. The wp_commentmeta table can store additional information about each comment

wp_user and wp_usermeta

1. The wp_users table stores all the information about your website's users
2. Each row in the wp_users table represents a single user
3. Like other meta tables, the wp_usermeta table can store additional information about each user

wp_terms, wp_termmeta, wp_term_relationships, wp_term_taxonomy

1. Manage the categories and tags (“terms”)
2. What determines whether they are a category or a tag is the taxonomy that they are associated with, which is stored in wp_term_taxonomy
3. The wp_term_relationships table stores the relationships between terms and their parent objects, be that a post, page, or custom post type.



wp_options

1. Each row in the wp_options table represents a specific WP setting
2. The wp_options table also stores information about your website's active theme and active plugins
3. The Options API provides functions to interact: add_option, update_option, delete_option

WordPress DB API

1. Insert, update, delete e.g. *wp_insert_post*, *wp_update_post*, *wp_delete_post*
2. Fetch records e.g. *get_post*, *get_posts*
3. Insert, update, or delete meta fields e.g. *add_post_meta*, *update_post_meta*, *delete_post_meta*

WC Tables

Meet the tables behind the magic

Products, Orders, Customers, but also...

1. Action Scheduler
2. Sessions
3. Order items
4. Tax rates
5. Shipping rates
6. Payments
7. Logs
8. Webhooks
9. Lookups
10. And HPOS tables!

Core WooCommerce Tables 1/2

1. woocommerce_sessions - Customer session data (carts, etc.)
2. woocommerce_api_keys - REST API keys
3. woocommerce_attribute_taxonomies - Global product attribute taxonomies
4. woocommerce_downloadable_product_permissions - Download permissions for products
5. woocommerce_order_items - Order line items (legacy)
6. woocommerce_order_itemmeta - Order line item meta data (legacy)
7. woocommerce_tax_rates - Tax rates

Core WooCommerce Tables 2/2

8. woocommerce_tax_rate_locations - Tax rate locations (postcodes/cities)
9. woocommerce_shipping_zones - Shipping zones
10. woocommerce_shipping_zone_locations - Shipping zone locations
11. woocommerce_shipping_zone_methods - Shipping zone methods
12. woocommerce_payment_tokens - Customer payment tokens
13. woocommerce_payment_tokenmeta - Payment token meta data
14. woocommerce_log - General logging table

WooCommerce Tables with "wc_" prefix

15. wc_webhooks - Webhooks
16. wc_download_log - Download logs
17. wc_product_meta_lookup - Product meta lookup/indexing
18. wc_tax_rate_classes - Tax classes
19. wc_reserved_stock - Reserved stock (prevents checkout race conditions)

HPOS Tables

20. wc_orders - Main orders table
21. wc_order_addresses - Order shipping/billing addresses
22. wc_order_operational_data - Order operational data
23. wc_orders_meta - Order meta data

WooCommerce Admin/Analytics Tables

24. wc_order_stats - Order statistics for analytics
25. wc_order_product_lookup - Order-product lookup for reporting
26. wc_order_tax_lookup - Order tax lookup for reporting
27. wc_order_coupon_lookup - Order coupon lookup for reporting
28. wc_admin_notes - Admin inbox notifications
29. wc_admin_note_actions - Admin note actions
30. wc_customer_lookup - Customer lookup for analytics
31. wc_category_lookup - Category lookup for analytics

Action Scheduler Tables

32. actionscheduler_actions - Scheduled actions
33. actionscheduler_claims - Action claims
34. actionscheduler_groups - Action groups
35. actionscheduler_logs - Action execution logs

Clearing the Confusion

1. **Orders: the heartbeat of your store**
2. **Products: the items you sell**
3. **Users/Customers: who you serve**

Most queries, reports, and plugin features revolve around these three. By concentrating here first, you avoid unnecessary complexity and confusion from the dozens of other tables you rarely touch.

Order Tables

How order data is truly stored…
since HPOS

Orders Tables Overview (HPOS)

1. wp_wc_orders
2. wp_wc_orders_meta
3. wp_wc_order_addresses
4. wp_wc_order_coupon_lookup
5. wp_wc_order_operational_data
6. wp_wc_order_product_lookup
7. wp_wc_order_stats
8. wp_wc_order_tax_lookup

wp_wc_orders

1. id — Primary key, bigint(20) UNSIGNED
2. status — Order status, varchar(20)
3. currency — Currency code, varchar(10)
4. type — Order type, varchar(20)
5. tax_amount — Total tax, decimal(26,8)
6. total_amount — Total order amount, decimal(26,8)
7. customer_id — Customer reference, bigint(20) UNSIGNED
8. billing_email — Customer billing email, varchar(320)
9. date_created_gmt — Order creation date (GMT), datetime

wp_wc_orders

10. date_updated_gmt — Order update date (GMT), datetime
11. parent_order_id — Parent order reference, bigint(20) UNSIGNED
12. payment_method — Payment method ID, varchar(100)
13. payment_method_title — Payment method name/title, text
14. transaction_id — Payment transaction ID, varchar(100)
15. ip_address — Customer IP address, varchar(100)
16. user_agent — Customer browser info, text
17. customer_note — Notes left by the customer, text

wp_wc_orders

id	status	currency	type	tax_amount	total_amount	customer_id	billing_e
18814	wc-completed	EUR	shop_order	0.00000000	145.00000000	4	cn
18815	wc-completed	EUR	shop_order	0.00000000	145.00000000	0	joh
18830	wc-completed	EUR	shop_order	0.00000000	145.00000000	0	ph
18835	wc-completed	EUR	shop_order	0.00000000	145.00000000	0	ma
18846	wc-completed	EUR	shop_order	0.00000000	145.00000000	0	ca

wp_wc_orders_meta

1. id — Primary key, bigint(20) UNSIGNED AUTO_INCREMENT
2. order_id — Foreign key to the main order, bigint(20) UNSIGNED
3. meta_key — Metadata key name, varchar(255)
4. meta_value — Metadata value, text

The meta table is where WooCommerce stores extra order info that is not in the main table.

wp_wc_orders_meta

id	order_id	meta_key	meta_value
1	18814	_mc4wp_optin	1
2	18814	_stripe_customer_id	cus_[redacted]
3	18814	_stripe_charge_id	ch_16[redacted]
4	18814	Stripe Payment ID	ch_16[redacted]
5	18814	Stripe Fee	4.58
6	18814	Net Revenue From Stripe	140.42
7	18814	_stripe_charge_captured	yes
8	18814	_paid_date	2015-07-10 19:16:43
9	18814	_order_shipping_base_currency	
10	18814	_cart_discount_base_currency	0
11	18814	_order_tax_base_currency	0
12	18814	_order_shipping_tax_base_currency	0
13	18814	_order_total_base_currency	145.00

wp_wc_order_addresses

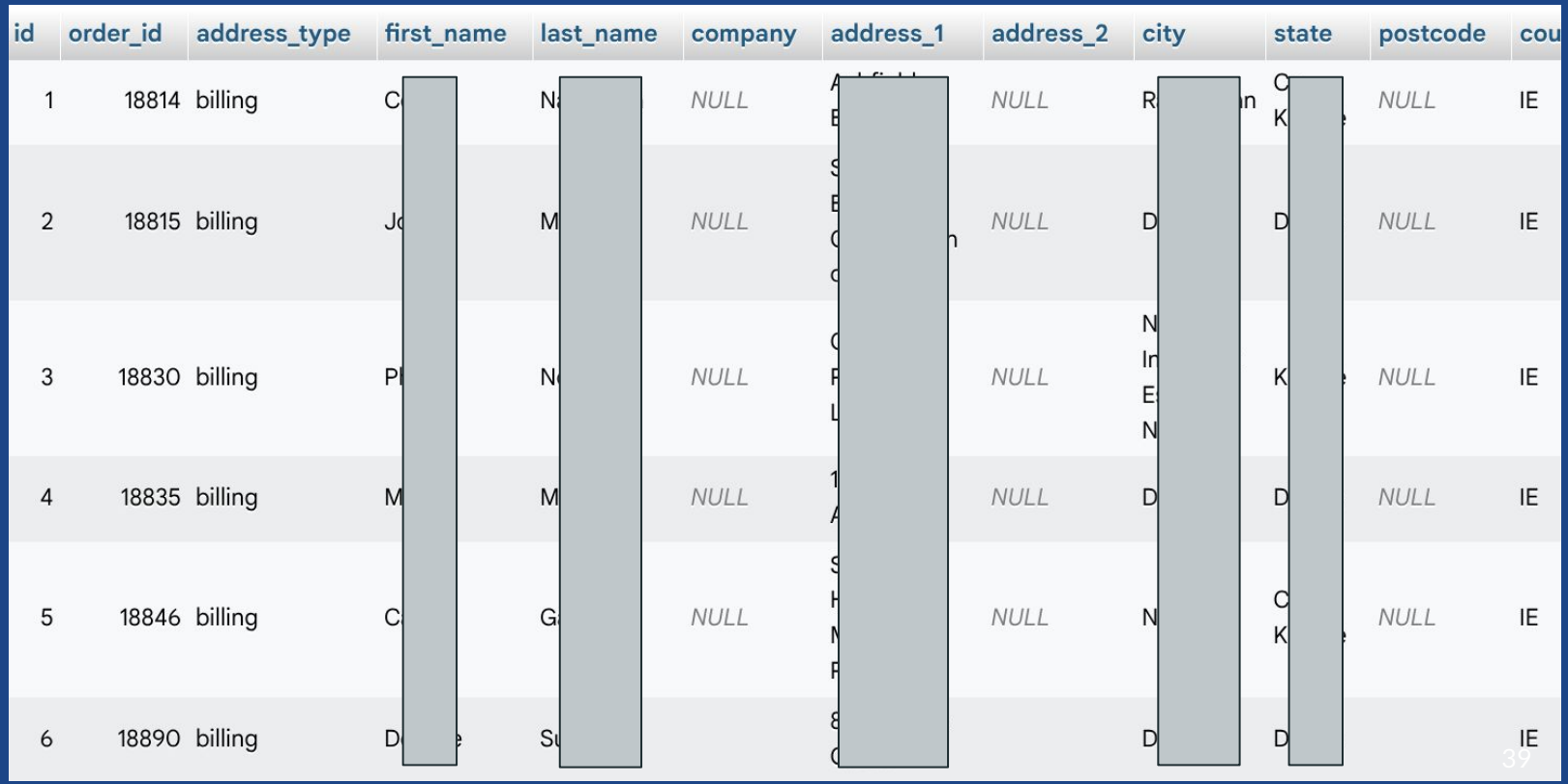
1. id — Primary key, bigint(20) UNSIGNED AUTO_INCREMENT
2. order_id — Foreign key to the order, bigint(20) UNSIGNED
3. address_type — Type of address, varchar(20)
4. first_name — Customer first name, text
5. last_name — Customer last name, text
6. company — Company name, text
7. address_1 — Main street address, text
8. address_2 — Additional address info, text

wp_wc_order_addresses

9. city — City, text
10. state — State/region, text
11. postcode — ZIP/postal code, text
12. country — Country code or name, text
13. email — Customer email, varchar(320)
14. phone — Customer phone number, varchar(100)

This table separates addresses from orders for performance and clarity. One order can have multiple addresses (billing vs. shipping).

wp_wc_order_addresses



id	order_id	address_type	first_name	last_name	company	address_1	address_2	city	state	postcode	cou
1	18814	billing	C	N	NULL	A B	NULL	R n	C K	NULL	IE
2	18815	billing	J	M	NULL	S B C c h	NULL	D	D	NULL	IE
3	18830	billing	P	N	NULL	C P L	NULL	N In E N	K	NULL	IE
4	18835	billing	M	M	NULL	1 A	NULL	D	D	NULL	IE
5	18846	billing	C	G	NULL	S H M P	NULL	N	C K	NULL	IE
6	18890	billing	D e	Su		8 C		D	D		IE

wp_wc_order_coupon_lookup

order_id	coupon_id	date_created	discount_amount
242688	242644	2023-12-20 01:44:10	447.75
242752	222093	2023-12-24 11:27:56	334.32

wp_wc_order_operational_data

id	order_id	created_via	woocommerce_version	prices_include_tax	coupon_usages_are_counted
1	18814	checkout	7.6.0	0	1
2	18815	checkout	7.6.0	0	1
3	18830	checkout	7.6.0	0	1
4	18835	checkout	7.6.0	0	1
5	18846	checkout	7.6.0	0	1

wp_wc_order_product_lookup

1. order_item_id — Primary key, bigint(20) UNSIGNED
2. order_id — Foreign key to the order, bigint(20) UNSIGNED
3. product_id — Product reference, bigint(20) UNSIGNED
4. variation_id — Variation reference, bigint(20) UNSIGNED
5. customer_id — Customer reference, bigint(20) UNSIGNED
6. date_created — Date/time of purchase, datetime
7. product_qty — Quantity purchased, int(11)
8. product_net_revenue — Net revenue, double

wp_wc_order_product_lookup

9. product_gross_revenue — Gross revenue, double
10. coupon_amount — Discount applied, double
11. tax_amount — Tax charged for the product, double
12. shipping_amount — Shipping cost assigned to this product, double
13. shipping_tax_amount — Tax on shipping for this product, double

This table is optimized for reporting and analytics. It precomputes totals, taxes, and shipping per product line to avoid expensive joins on order meta. **Won’t work unless Analytics is active.**

wp_wc_order_product_lookup

order_item_id	order_id	product_id	variation_id	customer_id	date_created	product_qty	product_net_revenue
41664	183816	139864	0	1	2020-06-25 00:04:38	1	0
41665	183817	139864	0	2	2020-06-25 00:08:59	1	0
41666	183818	139864	0	2	2020-06-25 00:11:51	1	0
41667	183819	139864	0	2	2020-06-25 00:14:11	1	0
41668	183820	139864	0	3	2020-06-25 01:02:10	1	0
41669	183821	139864	0	4	2020-06-25 02:07:51	1	0
41670	183822	139864	0	5	2020-06-25 02:35:30	1	0
41671	183823	139864	0	6	2020-06-25 03:03:41	1	0
41672	183824	139864	0	7	2020-06-25 06:17:47	1	0
41673	183825	139864	0	8	2020-06-25 06:25:51	1	0
41674	183826	139864	0	8	2020-06-25 06:29:45	1	0
41675	183827	139864	0	9	2020-06-25 07:17:59	1	0

wp_wc_order_tax_lookup

order_id	tax_rate_id	date_created	shipping_tax	order_tax	total_tax
242607	154	2023-12-15 11:29:16	0	0	0
242632	141	2023-12-17 21:59:58	0	0	0
242633	154	2023-12-18 16:37:26	0	0	0
242635	141	2023-12-18 20:41:49	0	0	0
242641	154	2023-12-19 10:30:16	0	0	0
242645	141	2023-12-19 10:49:24	0	0	0
242647	124	2023-12-19 11:03:57	0	0	0
242648	141	2023-12-19 12:22:34	0	0	0
242653	154	2023-12-19 15:40:12	0	0	0
242671	154	2023-12-19 22:25:25	0	0	0

Product Tables

Where products live and perform best

Product Tables Overview

1. wp_posts (WP) → Stores products with post_type = ‘product’ or ‘product_variation’
2. wp_postmeta (WP) → Stores metadata for products: prices, stock, SKU, visibility, dimensions, custom fields
3. wp_wc_product_meta_lookup → Lookup table for optimized queries: price, stock, dimensions, categories, attributes
4. wp_wc_product_attributes_lookup
5. wp_wc_product_download_directories

No HPPS for Products?

1. Orders used to live in wp_posts and wp_postmeta, which made queries painfully slow at scale. HPOS was introduced to fix this—giving orders their own normalized tables.
2. **Orders are constantly inserted and updated; products, on the other hand, are relatively static**. That means the traditional post/meta model works fine in most stores.
3. Product data performance is boosted by wp_wc_product_meta_lookup, which handles reporting, filtering, and search efficiently

wp_wc_product_meta_lookup

1. Purpose: performance-focused product data
2. *product_id*: links to wp_posts.ID
3. *sku*: quick SKU-based lookups
4. *min_price / max_price*: useful for filtering variable products
5. *stock_status*, *onsale*: enable fast catalog filtering
6. *average_rating*, *total_sales*: precomputed stats
7. Instead of querying dozens of postmeta rows, you can grab stock, price, and rating info instantly from this lookup table. Perfect for custom product loops, reports, or sorting logic.

wp_wc_product_meta_lookup

product_id	sku	virtual	downloadable	min_price	max_price	onsale	stock_quantity	stock_status
17210		1	0	199.0000	199.0000	0	3	instock
18715		1	0	145.0000	145.0000	1	0	outofstock
18858	NULL	0	0	295.0000	295.0000	0	NULL	NULL
18882		1	0	145.0000	145.0000	1	16	instock
18953		1	0	145.0000	145.0000	1	10	instock
19090	NULL	0	0	78.8600	78.8600	0	NULL	NULL
19157	NULL	0	0	395.0000	395.0000	0	NULL	NULL
19160		0	0	150.0000	650.0000	0	NULL	instock
19731	starter	1	0	497.0000	497.0000	0	NULL	instock
19732	pro	1	0	795.0000	795.0000	0	NULL	instock
19733	business	0	0	810.5700	810.5700	0	NULL	instock
19790		1	0	15.4500	15.4500	0	NULL	instock

Queries

Turning database insights into real solutions

Querying WooCommerce Data Safely

1. **Don’t jump straight to SQL**

Direct queries can break with updates, miss filters, or bypass caching and hooks. They’re powerful but fragile.

2. **Use the CRUD (Create-Read-Update-Delete) layer instead**

There are WooCommerce PHP functions that let you perform operations safely without writing raw SQL.

3. Using CRUD ensures data integrity, hooks/triggers fire correctly, and future updates won’t break your code.

Querying Orders via PHP

wc_get_order() to fetch 1 order

```
// Get order object
$order = wc_get_order( 123 );

// And then...
$order->get_currency();
$order->get_total();
$order->get_customer_id();
$order->get_shipping_country();
```

wc_get_orders() to fetch N orders

```
// Get orders processing
$orders = wc_get_orders( array(
'status' => array( 'wc-processing' ) )
);

// Get orders from the US
$orders = wc_get_orders( array(
'billing_country' => 'US' ) );
```

Why wc_get_orders() Beats SQL

1. **Future-proof against schema changes** (hello, HPOS)
2. **Handles hooks, caching, and internal logic** (SQL bypasses these)
3. **Simplifies complex queries** (no need to join multiple tables)
4. **Reduces errors and security risks** (direct SQL increases the chance of typos, missing conditions, or SQL injection)
5. **Works with extensions and future features** (plugins and WooCommerce core rely on CRUD objects, not raw tables)

Querying Products via PHP

wc_get_product() for 1 product

```
// Get product object
$product = wc_get_product( 123 );

// And then…
$product->get_name();
$product->get_price();

$product->set_weight();
$product->save();
```

wc_get_products() for N products

```
$products = wc_get_products([
   'limit' => 10,
   'category' => ['t-shirts'],
   'stock_status' => 'instock',
]);
```

Querying Customers via PHP

WC_Customer() for 1 user

```
// Get customer object
$customer = new WC_Customer( $user_id );

// And then…
$email = $customer->get_email();
$first_name = $customer->get_first_name();
```

WC_Customer_Query() for N users

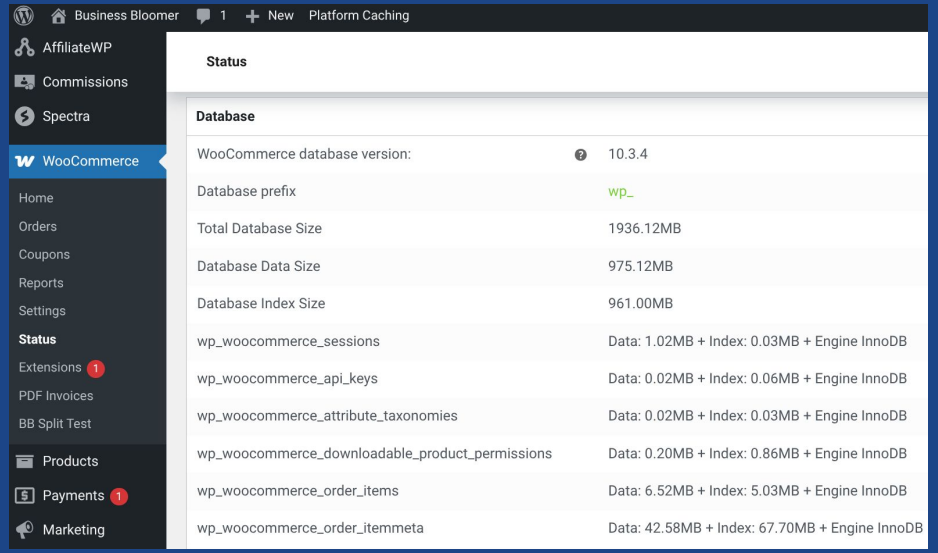
```
$query = new WC_Customer_Query([
   'role' => 'customer',
   'orderby' => 'date_registered',
   'order' => 'DESC',
]);
$customers = $query->get_results();
```

Maintenance

Keep your database lean and fast

WooCommerce System Status

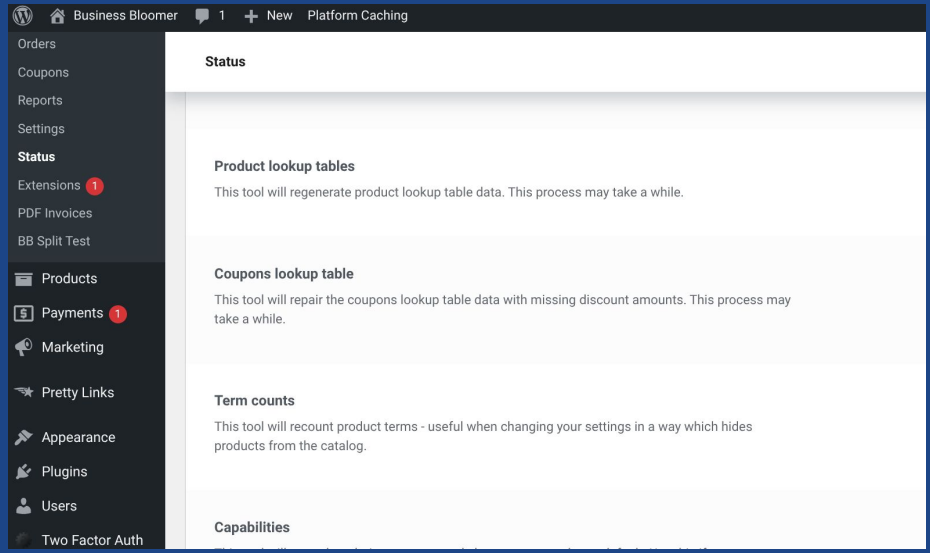
1. Provides a snapshot of your WooCommerce environment, including WordPress, PHP, **database**, server, and plugin info.
2. Database: check periodically for unusually large tables!



wp_actionscheduler_logs Data: 114.66MB + Index: 73.84MB + Engine InnoDB

WooCommerce Status > Tools

1. Offers quick actions to rebuild, clean, or reset parts of the store without manually touching the database.
2. Regenerate lookup tables
3. Recount terms
4. Clear transients
5. Rebuild order lookup tables
6. Delete expired session



WooCommerce Status > Useful DB Tools

1. **WooCommerce transients**: clears product/shop transient cache stored in the database for faster queries.
2. **Orphaned variations**: removes variations that no longer have a parent.
3. **Product lookup tables**: regenerates wp_wc_product_meta_lookup data for optimized product queries.
4. **Clear customer sessions**: deletes all customer session data, including carts saved in the database.
5. **Verify base database tables**: do all core WooCommerce tables exist?
6. **Clean up order data from legacy tables**: deletes legacy tables.

⚠️ DANGER ZONE ALERT ⚠️

1. Only do this if you know what you're doing
2. Only do this if you're 1000% sure the data you're about to edit/delete is the CORRECT data
3. Avoid running on a live site unless necessary
4. Back up before anything
5. This section is about true "danger zone" operations – high risk, high reward

Clean Action Scheduler Log

Stores logs for every Action Scheduler job: status, messages, errors. Deleting old logs will not break WooCommerce, but you lose historical job info.

```
TRUNCATE TABLE wp_actionscheduler_logs;
```

wp_actionscheduler_logs	Data: 114.66MB + Index: 73.84MB + Engine InnoDB

Trim Orphaned wp_postmeta

wp_postmeta stores metadata for every post, product, order, or custom post type. Orphaned rows occur when a post/product/order is deleted, but its metadata remains.

```
SELECT COUNT(*) FROM wp_postmeta pm
LEFT JOIN wp_posts p ON pm.post_id = p.ID
WHERE p.ID IS NULL;
```

COUNT(*)
1084

```
DELETE pm FROM wp_postmeta pm
LEFT JOIN wp_posts p ON pm.post_id = p.ID
WHERE p.ID IS NULL;
```

Clean Up Autoloaded Data

Options with autoload = yes are loaded on every page request. If too many large options are autoloaded, it can increase page load time.

```
SELECT option_name,
LENGTH(option_value)/1024 AS size_kb
FROM wp_options
WHERE autoload='yes'
ORDER BY size_kb DESC
LIMIT 20;
```

option_name	size_kb ▾ 1
aruba_fe_wc_tax_rates_backup	111.5371
ast-block-templates-customizer-css	28.8125
rewrite_rules	27.8174
limit_login_logged	23.0439
affwp_settings	12.2090
wpseo_titles	10.0703
woocommerce_tracker_ua	6.8242
option_tree	5.7275
wpseo	5.3506
stats_cache	5.1592
get_spectra_block_count	4.7715
newsletter_subscription_lists	4.7676
newsletter_profile	4.6650

Conclusion

Key lessons, resources, and closing thoughts

WooCommerce Database Tables

Now you should understand the WooCommerce database data better:

- You know which tables matter most (orders, products, customers) and which are mostly legacy.
- You understand how WooCommerce stores metadata and lookup data.
- You're aware of tools and SQL strategies to safely maintain, clean, and optimize the database.
- You're equipped to query, debug, and extend WooCommerce with confidence, without relying on guesswork.