

Product Management

1. Get all products:

SELECT * FROM products;

The screenshot shows the phpMyAdmin interface for a MySQL database named 'database'. The 'Table: products' page is displayed. The results of the query 'SELECT * FROM products;' are shown in a table with 4 rows. The columns are: product_id, product_name, description, regular_price, sale_price, stock_quantity, sku, weight, dimensions, created_at, updated_at, and is_active.

product_id	product_name	description	regular_price	sale_price	stock_quantity	sku	weight	dimensions	created_at	updated_at	is_active
2	first aid kits	Class A and Class B	2500.00	1500.00	10	ANSI A+	650.00	225 x 235 x 95mm	2025-06-30 10:35:09	2025-06-30 10:35:09	1
3	White Casual Wears	a variety of comfortable and relaxed clothing opti...	500.00	250.00	55	BLM-A101	156.00	T-S 38.0 24.0 M 40.0 26.0 16.0 17.0 L 42.0 28.0 ...	2025-06-30 10:35:09	2025-06-30 11:03:14	1
4	New Gadget	A cool new device.	99.99	NULL	100	NULL	NULL	NULL	2025-06-30 11:00:48	2025-06-30 11:00:48	1
1	PAPER	A cool new device.	100.00	NULL	54	NULL	NULL	NULL	2025-06-30 11:00:48	2025-06-30 21:25:23	1

2. Get product by ID:

SELECT * FROM products WHERE product_id = 1

The screenshot shows the phpMyAdmin interface for a MySQL database named 'database'. The 'Table: products' page is displayed. The results of the query 'SELECT * FROM products WHERE product_id = 1;' are shown in a table with 1 row. The columns are: product_id, product_name, description, regular_price, sale_price, stock_quantity, sku, weight, dimensions, created_at, updated_at, and is_active.

product_id	product_name	description	regular_price	sale_price	stock_quantity	sku	weight	dimensions	created_at	updated_at	is_active
1	PAPER	A cool new device.	100.00	NULL	54	NULL	NULL	NULL	2025-06-30 11:00:48	2025-06-30 21:25:23	1

3. Get products by category:

SELECT p.* FROM products p JOIN product_categories pc ON

p.product_id = pc.product_id WHERE pc.category_id = 5;

The screenshot shows the phpMyAdmin interface for a MySQL database named 'database'. The current table is 'products'. A warning message at the top states: 'Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.' Below this, a green bar indicates 'Showing rows 0 - 0 (1 total, Query took 0.0009 seconds)'. The SQL query is:

```
SELECT p.* FROM products p JOIN product_categories pc ON p.product_id = pc.product_id WHERE pc.category_id = 5;
```

The results table has the following columns: product_id, product_name, description, regular_price, sale_price, stock_quantity, sku, weight, dimensions, created_at, updated_at, and is_active. One row is shown:

product_id	product_name	description	regular_price	sale_price	stock_quantity	sku	weight	dimensions	created_at	updated_at	is_active
1	PAPER	A cool new device.	100.00	NULL	5	NULL	NULL	NULL	2025-06-30 11:00:48	2025-07-01 10:57:20	1

4. Get products with low stock (e.g., less than 10):

SELECT * FROM products WHERE stock_quantity < 10

The screenshot shows the phpMyAdmin interface for a MySQL database named 'database'. The current table is 'products'. A green bar indicates 'Showing rows 0 - 0 (1 total, Query took 0.0051 seconds.)'. The SQL query is:

```
SELECT * FROM products WHERE stock_quantity < 10;
```

The results table has the following columns: product_id, product_name, description, regular_price, sale_price, stock_quantity, sku, weight, dimensions, created_at, and updated_at. One row is shown:

product_id	product_name	description	regular_price	sale_price	stock_quantity	sku	weight	dimensions	created_at	updated_at
1	first aid kits	Class A and Class B	2500.00	1500.00	5	ANSI	650.00	225 x 235 x 95mm	2025-06-30 10:35:09	2025-07-01 14:59:17

5. Get products on sale:

SELECT * FROM products WHERE sale_price IS NOT NULL AND sale_price < regular_price

Showing rows 0 - 1 (2 total, Query took 0.0036 seconds.)

```
SELECT * FROM products WHERE sale_price IS NOT NULL AND sale_price < regular_price;
```

product_id	product_name	description	regular_price	sale_price	stock_quantity	sku	weight	dimensions	created_at	updated_at	is_active
2	first aid kits	Class A and Class B	2500.00	1500.00	5	ANSI A+	650.00 95mm	225 x 235 x 10.35.09	2025-06-30	2025-07-01	1
3	White Casual Wears	a variety of comfortable and relaxed clothing opt...	500.00	250.00	55	TS-BLM-A101	156.00 17.0	M 40.0 26.0 L 42.0 28.0	2025-06-30	2025-06-30	1

6.Add a new product:

INSERT INTO products (product_name, description, regular_price,

stock_quantity, created_at)

VALUES ('New Gadget', 'A cool new device.', 99.99, 100, NOW());

1 row inserted.
Inserted row id: 6 (Query took 0.0006 seconds.)

```
INSERT INTO products (product_name, description, regular_price, stock_quantity, created_at) VALUES ('New Gadget', 'A cool new device.', 99.99, 100, NOW());
```

7.Update product price:

UPDATE products SET regular_price = 109.99 WHERE product_id = 1

The screenshot shows the phpMyAdmin interface for a MySQL database named 'database'. The left sidebar lists various tables: New, database, New, cart_items, categories, coupons, coupon_usages, orders, order_items, payments, products, product_attributes, product_attribute_values, product_categories, product_images, product_reviews, product_views, refunds, shipping_methods, and users. The main area is titled 'Table: products' and contains a SQL query editor. The query is:

```
1 UPDATE products SET regular_price = 109.99 WHERE product_id = 1
```

The results pane on the right shows the columns for the products table: product_id, product_name, description, regular_price, sale_price, stock_quantity, sku, weight, dimensions, created_at, updated_at, and is_active. Below the query editor are several buttons: SELECT *, SELECT, INSERT, UPDATE, DELETE, Clear, Format, Get auto-saved query, Bind parameters, Delimiter :, Show this query here again (checked), Retain query box, Rollback when finished, Enable foreign key checks, Simulate query, and Go.

8.Update product stock:

UPDATE products SET stock_quantity = 50 WHERE product_id = 1;

The screenshot shows the phpMyAdmin interface after executing the update query. The results pane displays a green message: '✓ 1 row affected. (Query took 0.0007 seconds.)'. The query shown is:

```
UPDATE products SET stock_quantity = 50 WHERE product_id = 1;
```

The left sidebar shows the same list of tables as the previous screenshot. The main area has a 'Show query box' button and three options below it: Edit inline, Edit, and Create PHP code.

9.Delete a product:

DELETE FROM products WHERE product_id = 1

The screenshot shows the phpMyAdmin interface for a MySQL database named 'database'. The left sidebar lists various tables: New, database, cart_items, categories, coupons, coupon_usages, orders, order_items, payments, products, product_attributes, product_attribute_values, product_categories, product_images, product_reviews, product_views, refunds, shipping_methods, users, user_addresses, and wishlists. The right panel displays the results of a DELETE query: 'DELETE FROM products WHERE product_id = 1;'. A message at the top indicates '1 row deleted. (Query took 0.0006 seconds.)'.

10. Search products by keyword (e.g., 'shirt'):

```
SELECT * FROM products WHERE product_name LIKE '%shirt%' OR
description LIKE '%shirt%'
```

The screenshot shows the phpMyAdmin interface for a MySQL database named 'database'. The left sidebar lists the same tables as the previous screenshot. The right panel displays the results of the search query: 'SELECT * FROM products WHERE product_name LIKE "%shirt%" OR description LIKE "%shirt%"'. It shows one result row for a product with product_id 3, which is a 'White Casual Wears' item. The table includes columns for product_id, product_name, description, regular_price, sale_price, stock_quantity, sku, weight, dimensions, created_at, updated_at, and is_active.

product_id	product_name	description	regular_price	sale_price	stock_quantity	sku	weight	dimensions	created_at	updated_at	is_active
3	White Casual Wears	a variety of comfortable and relaxed clothing opti...	500.00	250.00	55	TS- BLM-A101	16.0 17.0	S 38.0 24.0 L 42.0 28.0 ...	2025-06-30	2025-06-30 10:35:09	1

11. Get product images for a product:

```
SELECT * FROM product_images WHERE product_id = 1
```

The screenshot shows the phpMyAdmin interface for a MySQL database. The left sidebar lists various tables: New, database, New, cart_items, categories, coupons, coupon_usages, orders, order_items, payments, products, product_attributes, product_attribute_values, product_categories, product_images, product_reviews, product_views, refunds, shipping_methods, users, and user_addresses. The main area is titled 'Table: product_images'. It shows a single row with the following data:

image_id	product_id	image_url	alt_text	is_thumbnail	sort_order
1	1	http://example.com/image1.jpg	NULL	1	0

Below the table, there are buttons for Edit, Copy, Delete, and Export.

12.Add a new product image:

INSERT INTO product_images (product_id, image_url, is_thumbnail)

VALUES (1, 'http://example.com/image1.jpg', TRUE)

The screenshot shows the phpMyAdmin interface after an insertion operation. The message bar at the top indicates '1 row inserted.' and 'Inserted row id: 2 (Query took 0.0018 seconds.)'. The SQL query shown is:

```
INSERT INTO product_images (product_id, image_url, is_thumbnail) VALUES (1, 'http://example.com/image1.jpg', TRUE);
```

The table structure and data are identical to the previous screenshot, showing one row with image_id 1, product_id 1, image_url 'http://example.com/image1.jpg', alt_text NULL, is_thumbnail 1, and sort_order 0.

13.Get all categories:

SELECT * FROM categories

Showing rows 0 - 5 (6 total, Query took 0.0006 seconds.)

```
SELECT * FROM categories;
```

	category_id	category_name	description	parent_category_id
Edit	1	Electronics	traditional notebooks, gaming laptops, workstation...	0
Edit	2	Health Care	encompasses a wide range of services and industrie...	1
Edit	3	Fashion	casual wear, formal wear, ethnic wear, sportswear,...	2
Edit	4	Furniture	categorized by its function, style, and the materi...	3
Edit	5	Smartphones	action figures, building sets, dolls, educational ...	4
Edit	6	Electricals	NULL	NULL

14.Add a new category:

INSERT INTO categories (category_name) VALUES ('Electronics')

1 row inserted.
Inserted row id: 7 (Query took 0.0006 seconds.)

```
INSERT INTO categories (category_name) VALUES ('Electronics');
```

15.Update category name:

UPDATE categories SET category_name = 'Smartphones' WHERE

category_id = 5

The screenshot shows the phpMyAdmin interface for a MySQL database named 'database'. The left sidebar lists tables such as 'New', 'database', 'New', 'cart_items', 'categories', 'coupons', 'coupon_usages', 'orders', 'order_items', 'payments', 'products', 'product_attributes', 'product_attribute_values', 'product_categories', 'product_images', 'product_reviews', 'product_views', 'refunds', 'shipping_methods', 'users', 'user_addresses', and 'wishlist'. The main panel shows the results of an UPDATE query: 'UPDATE categories SET category_name = 'Smartphones' WHERE category_id = 5;'. A message at the top indicates '1 row affected. (Query took 0.0006 seconds.)'.

16.Delete a category:

DELETE FROM categories WHERE category_id = 5

The screenshot shows the phpMyAdmin interface for a MySQL database named 'database'. The left sidebar lists the same tables as the previous screenshot. The main panel shows the results of a DELETE query: 'DELETE FROM categories WHERE category_id = 5;'. A message at the top indicates '1 row deleted. (Query took 0.0046 seconds.)'.

17.Get products with their primary image:

```
SELECT p.*, (SELECT pi.image_url FROM product_images pi WHERE  
pi.product_id = p.product_id AND pi.is_thumbnail = TRUE LIMIT 1)  
AS thumbnail_url  
FROM products p
```

product_id	product_name	description	regular_price	sale_price	stock_quantity	sku	weight	dimensions	created_at	updated_at	is_active	thumbnail_url
2	first aid kits	Class A and Class B	2500.00	1500.00	5	ANSI A+	650.00	225 x 235 x 95mm	2025-06-30 10:35:09	2025-07-01 10:56:26	1	NULL
3	White Casual Wears	a variety of comfortable and relaxed clothing options...	500.00	250.00	55	TS-BLM-A101	156.00	M 40.0 26.0 17.0 L 42.0 28.0 ...	2025-06-30 10:35:09	2025-06-30 11:03:14	1	NULL
4	New Gadget	A cool new device.	99.99	NULL	100	NULL	NULL	NULL	2025-06-30 11:00:48	2025-06-30 11:00:48	1	NULL
6	New Gadget	A cool new device.	99.99	NULL	100	NULL	NULL	NULL	2025-07-01 11:13:43	2025-07-01 11:13:43	1	NULL

18. Get product attributes (e.g., size, color):

```
SELECT pa.attribute_name, pav.attribute_value
FROM product_attributes pa
JOIN product_attribute_values pav ON pa.attribute_id =
pav.attribute_id
WHERE pav.product_id = 1;
```

attribute_name	attribute_value
SBS	34677

19. Add a new product attribute value:

```
INSERT INTO product_attribute_values (product_id, attribute_id,
attribute_value)
VALUES (1, 10, 'Red');
```

The screenshot shows the phpMyAdmin interface. The left sidebar lists various database tables: New, database, New, cart_items, categories, coupons, coupon_usages, orders, order_items, payments, products, product_attributes, product_attribute_values, product_categories, product_images, product_reviews, product_views, refunds, shipping_methods, users, user_addresses, and wishlists. The main area shows a query result: "1 row inserted. Inserted row id: 2 (Query took 0.0006 seconds.)". Below it is the SQL query: "INSERT INTO product_attribute_values (product_id, attribute_id, attribute_value) VALUES (1, 10, 'Red');". There are buttons for Edit inline, Edit, Create PHP code, and Refresh.

20. Get top 10 most viewed products:

```
SELECT product_id, COUNT(*) AS view_count
FROM product_views
GROUP BY product_id
ORDER BY view_count DESC
LIMIT 10;
```

The screenshot shows the phpMyAdmin interface. The left sidebar lists the same set of tables as the previous screenshot. The main area displays the results of the executed query: "Your SQL query has been executed successfully." Below the message is the SQL query: "SELECT product_id, COUNT(*) AS view_count FROM product_views GROUP BY product_id ORDER BY view_count DESC LIMIT 10;". The results table shows one row: product_id 11 and view_count 1. At the bottom, there are options for Print, Copy to clipboard, Export, Display chart, and Create view.

21. User & Authentication

1. Register a new user:

```
INSERT INTO users (username, email, password_hash, created_at)
```

`VALUES ('john_doe', 'john@example.com', 'hashed_password_here',`

`NOW())`

The screenshot shows the phpMyAdmin interface for MySQL. The left sidebar lists various database tables. The main area shows a query result for an INSERT operation into the 'users' table. The message indicates 1 row inserted, with a note about data truncation for the 'created_at' column. The SQL query is:

```
INSERT INTO users (username, email, password_hash, created_at) VALUES ('john_doe', 'john@example.com', 'hashed_password_here', NOW());
```

22.login a user (retrieve password hash for verification):

`SELECT user_id, password_hash FROM users WHERE email =`

`'john@example.com'`

The screenshot shows the phpMyAdmin interface for MySQL. The left sidebar lists various database tables. The main area shows the results of a SELECT query on the 'users' table, filtering by email. The results table displays one row with columns 'user_id' and 'password_hash'. The SQL query is:

```
SELECT user_id, password_hash FROM users WHERE email = 'john@example.com';
```

23.Get user profile by ID:

`SELECT * FROM users WHERE user_id = 1;`

Showing rows 0 - 0 (1 total, Query took 0.0007 seconds.)

```
SELECT * FROM users WHERE user_id = 1;
```

	user_id	username	email	password_hash	first_name	last_name	phone_number	registration_date	last_login	is_active	role
	1	john_doe	john@example.com	hashed_password_here	NULL	NULL	NULL	2025-07-01 12:12:52	NULL	1	customer

24. Update user profile information:

UPDATE users SET first_name = 'John', last_name = 'Doe' WHERE

user_id = 1

1 row affected. (Query took 0.0049 seconds.)

```
UPDATE users SET first_name = 'John', last_name = 'Doe' WHERE user_id = 1;
```

25. Change user password:

UPDATE users SET password_hash = 'new_hashed_password' WHERE

user_id = 1

A screenshot of the phpMyAdmin interface. The left sidebar shows a tree view of database tables under the 'database' node, including 'New', 'cart_items', 'categories', 'coupons', 'coupon_usages', 'orders', 'order_items', 'payments', 'products', 'product_attributes', 'product_attribute_values', 'product_categories', 'product_images', 'product_reviews', 'product_views', 'refunds', 'shipping_methods', 'users', 'user_addresses', and 'wishlist'. The main query window shows a successful UPDATE operation:

```
UPDATE users SET password_hash = 'new_hashed_password' WHERE user_id = 1;
```

[Edit inline] [Edit] [Create PHP code]

26. Delete a user (soft delete recommended):

UPDATE users SET is_active = FALSE WHERE user_id = 1;

-- Or hard delete: **DELETE FROM users WHERE user_id = 1**

A screenshot of the phpMyAdmin interface, similar to the previous one but with two additional queries. The first query is an UPDATE, and the second is a DELETE:

```
UPDATE users SET is_active = FALSE WHERE user_id = 1;
```

[Edit inline] [Edit] [Create PHP code]

```
DELETE FROM users WHERE user_id = 1;
```

[Edit inline] [Edit] [Create PHP code]

27. Check if email exists:

SELECT COUNT(*) FROM users WHERE email = '%john@example.com%'

The screenshot shows the phpMyAdmin interface for a MySQL database. The left sidebar lists various tables: New, database, cart_items, categories, coupons, coupon_usages, orders, order_items, payments, products, product_attributes, product_attribute_values, product_categories, product_images, product_reviews, product_views, refunds, shipping_methods, users, user_addresses, and wishlists. The main area displays a query result for the 'users' table. The SQL query is:

```
SELECT email FROM users WHERE email LIKE '%john@example.com%';
```

The results show one row: `email` with value `john@example.com`. There are buttons for Edit, Copy, Delete, and Export.

28.Get all users (for admin panel):

SELECT * FROM users

The screenshot shows the phpMyAdmin interface for a MySQL database. The left sidebar lists various tables. The main area displays a query result for the 'users' table. The SQL query is:

```
SELECT * FROM users;
```

The results show two rows:

	user_id	username	email	password_hash	first_name	last_name	phone_number	registration_date	last_login	is_active	role	cre
1	1	john_doe	john@example.com	hashed_password	NULL	NULL	NULL	2025-07-01 12:12:52	NULL	1	customer	2025-07-01 12:12:52
2	2	john	sowmi@gmail.com	hashed_password	NULL	NULL	NULL	2025-07-01 12:12:52	NULL	1	customer	2025-07-01 12:12:52

29.Get user's shipping addresses:

SELECT * FROM user_addresses WHERE user_id = 1 AND address_type = 'shipping';

Click to go back, hold to see history

Server: MySQL:3306 > Database: database > Table: user_addresses

Browse Structure SQL Search Insert Export Import Privileges Operations Triggers

Show query box

Showing rows 0 - 0 (1 total, Query took 0.0006 seconds.)

SELECT * FROM user_addresses WHERE user_id = 1 AND address_type = 'shipping';

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

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

Extra options

address_id	user_id	address_line1	city	state	zip_code	country	address_type	is_default	
1	1	NADARMEDU	MODAKKURICHI	ERODE	TAMILNADU	YRG	INDIA	shipping	1

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

Query results operations

Print Copy to clipboard Export Display chart Create view

30.Add a new user address:

```
INSERT INTO user_addresses (user_id, address_line1, city, state,
zip_code, country, address_type)
VALUES (1, '123 Main St', 'Anytown', 'CA', '90210', 'USA',
'shipping')
```

Click to go back, hold to see history

Server: MySQL:3306 > Database: database > Table: user_addresses

Browse Structure SQL Search Insert Export Import Privileges Operations Triggers

Show query box

1 row inserted.

Inserted row id: 2 (Query took 0.0018 seconds.)

INSERT INTO user_addresses (user_id, address_line1, city, state, zip_code, country, address_type) VALUES (1, '123 Main St', 'Anytown', 'CA', '90210', 'USA', 'shipping');

[Edit inline] [Edit] [Create PHP code]

31.Update a user address:

```
UPDATE user_addresses SET address_line1 = '456 Oak Ave' WHERE
address_id = 1;
```

The screenshot shows the phpMyAdmin interface for MySQL. The left sidebar lists various database tables: New, database, cart_items, categories, coupons, coupon_usages, orders, order_items, payments, products, product_attributes, product_attribute_values, product_categories, product_images, product_reviews, product_views, refunds, shipping_methods, users, user_addresses, and wishlists. The main panel displays a SQL query window with the following content:

```
Server: MySQL:3306 > Database: database > Table: user_addresses
Show query box
1 row affected. (Query took 0.0006 seconds.)
UPDATE user_addresses SET address_line1 = '456 Oak Ave' WHERE address_id = 1;
[Edit inline] [Edit] [Create PHP code]
```

32.Delete a user address:

DELETE FROM user_addresses WHERE address_id = 1

The screenshot shows the phpMyAdmin interface for MySQL. The left sidebar lists various database tables. The main panel displays a SQL query window with the following content:

```
Server: MySQL:3306 > Database: database > Table: user_addresses
Show query box
1 row deleted. (Query took 0.0006 seconds.)
DELETE FROM user_addresses WHERE address_id = 1;
[Edit inline] [Edit] [Create PHP code]
```

33.Set a default shipping address for a user:

**UPDATE user_addresses SET is_default = FALSE WHERE user_id = 1 AND
address_type = 'shipping';**

UPDATE user_addresses SET is_default = TRUE WHERE address_id = 2

```

Server: MySQL:3306 > Database: database > Table: user_addresses
Browse Structure SQL Search Insert Export Import Privileges Operations Triggers
Show query box

✓ 2 rows affected. (Query took 0.0007 seconds.)
UPDATE user_addresses SET is_default = FALSE WHERE user_id = 1 AND address_type = 'shipping';
[Edit inline] [Edit] [Create PHP code]

✓ 1 row affected. (Query took 0.0006 seconds.)
UPDATE user_addresses SET is_default = TRUE WHERE address_id = 2;
[Edit inline] [Edit] [Create PHP code]

```

The screenshot shows the phpMyAdmin interface for a MySQL database named 'database'. The current table is 'user_addresses'. The left sidebar lists various tables: New, database, New, cart_items, categories, coupons, coupon_usages, orders, order_items, payments, products, product_attributes, product_attribute_values, product_categories, product_images, product_reviews, product_views, refunds, shipping_methods, and users. The main area displays two SQL queries. The first query updates a row where user_id is 1 and address_type is 'shipping' to set is_default to FALSE. The second query updates a row where address_id is 2 to set is_default to TRUE. Both queries were executed successfully.

34. Get default shipping address for a user:

```

SELECT * FROM user_addresses WHERE user_id = 1 AND address_type =
'shipping' AND is_default = TRUE

```

address_id	user_id	address_line1	address_line2	city	state	zip_code	country	address_type	is_default
2	1	123 Main St	NULL	Anytown	CA	90210	USA	shipping	1

The screenshot shows the results of the query from step 34. It displays one row in the 'user_addresses' table. The row has address_id 2, user_id 1, address_line1 '123 Main St', and is_default 1, indicating it is the default shipping address for user_id 1.

35.Shopping Cart & Wishlist

Add item to cart:

```

INSERT INTO cart_items (user_id, product_id, quantity)

```

```

VALUES (1, 10, 2)ON DUPLICATE KEY UPDATE quantity = quantity + VALUES(quantity); --

```

For updating quantity if item already exists

2 rows inserted. (Query took 0.0008 seconds.)

```
INSERT INTO cart_items (user_id, product_id, quantity) VALUES (1, 10, 2) ON DUPLICATE KEY UPDATE quantity = quantity + VALUES(quantity);
```

Warning: #1287 'VALUES function' is deprecated and will be removed in a future release. Please use an alias (INSERT INTO ... VALUES (...) AS alias) and replace VALUES(col) in the ON DUPLICATE KEY UPDATE clause with alias.col instead.

36. Get user's cart items:

```
SELECT ci.cart_item_id, p.product_name, p.regular_price,
ci.quantity, (p.regular_price * ci.quantity) AS subtotal
FROM cart_items ci
JOIN products p ON ci.product_id = p.product_id
WHERE ci.user_id = 1
```

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0067 seconds.)

```
SELECT ci.cart_item_id, p.product_name, p.regular_price, ci.quantity, (p.regular_price * ci.quantity) AS subtotal FROM cart_items ci JOIN products p ON ci.product_id = p.product_id WHERE ci.user_id = 1;
```

cart_item_id	product_name	regular_price	quantity	subtotal

37. Update cart item quantity:

```
UPDATE cart_items SET quantity = 3 WHERE cart_item_id = 5
```

The screenshot shows the phpMyAdmin interface for MySQL. The left sidebar lists databases and tables, including 'cart_items'. The main area shows a query result: '1 row affected. (Query took 0.0006 seconds.)' followed by the SQL command: 'UPDATE cart_items SET quantity = 3 WHERE cart_item_id = 5;'. Below the command are links for 'Edit inline', 'Edit', and 'Create PHP code'.

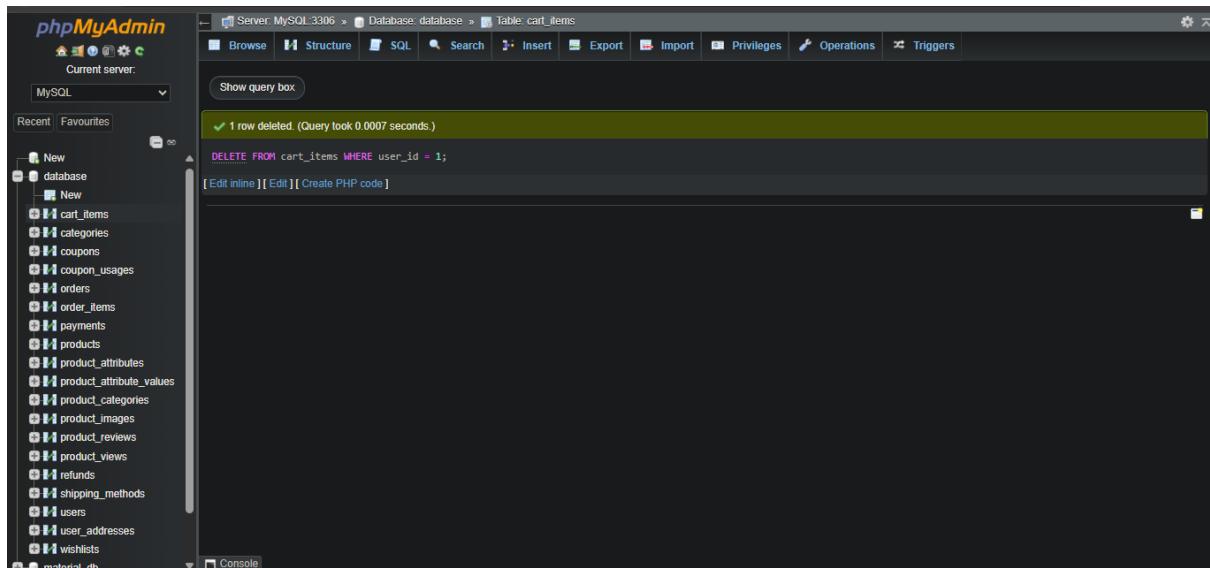
38.Remove item from cart:

DELETE FROM cart_items WHERE cart_item_id = 5;

The screenshot shows the phpMyAdmin interface for MySQL. The left sidebar lists databases and tables, including 'cart_items'. The main area shows a query result: '1 row deleted. (Query took 0.0028 seconds.)' followed by the SQL command: 'DELETE FROM cart_items WHERE cart_item_id = 5;'. Below the command are links for 'Edit inline', 'Edit', and 'Create PHP code'.

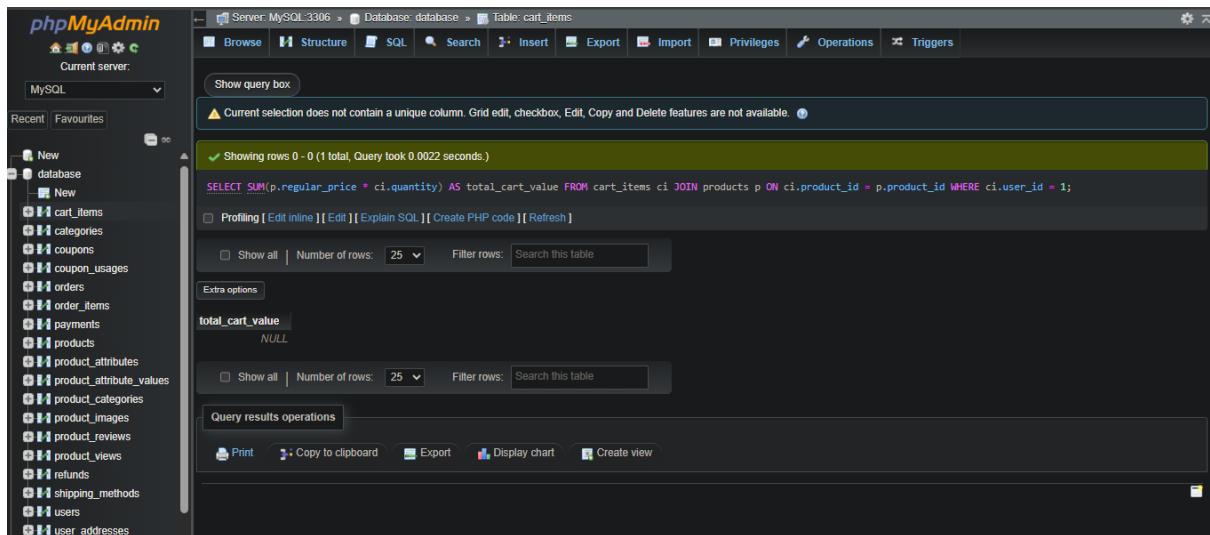
39.Clear user's cart:

DELETE FROM cart_items WHERE user_id = 1



40. Get total cart value for a user:

```
SELECT SUM(p.regular_price * ci.quantity) AS total_cart_value
FROM cart_items ci
JOIN products p ON ci.product_id = p.product_id
WHERE ci.user_id = 1
```



41. Add item to wishlist:

```
INSERT INTO wishlists (user_id, product_id, added_at)
VALUES (1, 15, NOW())
```

The screenshot shows the phpMyAdmin interface for a MySQL database. The left sidebar lists various tables under the 'database' section, including cart_items, categories, coupons, coupon_usages, orders, order_items, payments, products, product_attributes, product_attribute_values, product_categories, product_images, product_reviews, product_views, refunds, shipping_methods, users, user_addresses, and wishlists. The right panel is titled 'Table: cart_items' and shows a success message: '1 row inserted. Inserted row id: 1 (Query took 0.0068 seconds.)'. Below this, the SQL query is displayed: 'INSERT INTO wishlists (user_id, product_id, added_at) VALUES (1, 15, NOW());'. There are also links for 'Edit inline', 'Edit', and 'Create PHP code'.

42. Get user's wishlist items:

```
SELECT w.wishlist_id, p.product_name, p.regular_price, w.added_at
FROM wishlists w
JOIN products p ON w.product_id = p.product_id
WHERE w.user_id = 1
```

The screenshot shows the phpMyAdmin interface for a MySQL database. The left sidebar lists various tables under the 'database' section. The right panel is titled 'Table: wishlists' and displays the results of the executed SQL query. A warning message at the top states: 'Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.' The results table shows two rows of data:

wishlist_id	product_name	regular_price	added_at
1	New Gadget	99.99	2025-07-01 16:35:36
2	New Gadget	99.99	2025-07-01 16:35:36

43. Remove item from wishlist:

```
DELETE FROM wishlists WHERE wishlist_id = 3
```

The screenshot shows the phpMyAdmin interface for a MySQL database. The left sidebar lists tables such as cart_items, categories, coupons, etc. The main area shows a successful execution of a DELETE query:

```
DELETE FROM wishlists WHERE wishlist_id = 3;
```

Feedback message: ✓ 1 row deleted. (Query took 0.0006 seconds.)

44.Order Processing

Create a new order (after successful payment):

```
INSERT INTO orders (user_id, order_date, total_amount,
order_status, shipping_address_id)

VALUES (1, NOW(), 150.00, 'Pending', 1)
```

The screenshot shows the phpMyAdmin interface for a MySQL database. The left sidebar lists tables such as cart_items, categories, coupons, etc. The main area shows a successful execution of an INSERT query:

```
INSERT INTO orders (user_id, order_date, total_amount, order_status, shipping_address_id) VALUES (1, NOW(), 150.00, 'Pending', 1);
```

Feedback message: ✓ 1 row inserted. Inserted row id: 1 (Query took 0.0018 seconds.)

45.Add order items:

```
INSERT INTO order_items (order_id, product_id, quantity,
unit_price)

VALUES (10, 1, 2, 50.00), (10, 5, 1, 50.00);
```

```

Server: MySQL:3306 > Database: database > Table: orders
Browse Structure SQL Search Insert Export Import Privileges Operations Triggers
Show query box
2 rows inserted.
Inserted row id: 2 (Query took 0.0060 seconds.)
INSERT INTO order_items (order_id, product_id, quantity, unit_price) VALUES (10, 1, 2, 50.00), (10, 5, 1, 50.00);
[Edit inline] [Edit] [Create PHP code]

```

46. Get all orders for a user:

SELECT * FROM orders WHERE user_id = 1 ORDER BY order_date DESC;

order_id	user_id	order_date	total_amount	order_status	shipping_address_id	billing_address_id	tracking_number	shipped_date	delivered_date
1	1	2025-07-01 16:43:27	150.00	Pending	1	NULL	NULL	NULL	NULL

47. Get order details by order ID:

**SELECT o.*, ua.address_line1, ua.city, ua.state, ua.zip_code,
ua.country
FROM orders o
JOIN user_addresses ua ON o.shipping_address_id = ua.address_id
WHERE o.order_id = 10;**

Server: MySQL 3.306 > Database: database

Structure SQL Search Query Export Import Operations Privileges Routines Events Triggers Designer

Show query box

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0034 seconds.)

```
SELECT o.* , ua.address_line1 , ua.city , ua.state , ua.zip_code , ua.country FROM orders o JOIN user_addresses ua ON o.shipping_address_id = ua.address_id WHERE o.order_id = 10;
```

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

order_id | user_id | order_date | total_amount | order_status | shipping_address_id | billing_address_id | tracking_number | shipped_date | delivered_date | address_line1 | city | state | zip_code | country

Query results operations

Create view

The screenshot shows the phpMyAdmin interface with a sidebar containing a tree view of database tables. The main area displays a query result for order items where the order ID is 10. The result set is empty, as indicated by the message "MySQL returned an empty result set (i.e. zero rows). (Query took 0.0034 seconds.)". The query itself is shown in the SQL tab:

```
SELECT o.* , ua.address_line1 , ua.city , ua.state , ua.zip_code , ua.country FROM orders o JOIN user_addresses ua ON o.shipping_address_id = ua.address_id WHERE o.order_id = 10;
```

48. Get order items for a specific order:

```
SELECT oi.* , p.product_name
FROM order_items oi
JOIN products p ON oi.product_id = p.product_id
WHERE oi.order_id = 10
```

Server: MySQL 3.306 > Database: database

Structure SQL Search Query Export Import Operations Privileges Routines Events Triggers Designer

Show query box

Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.

Showing rows 0 - 0 (1 total). Query took 0.0021 seconds.

```
SELECT oi.* , p.product_name FROM order_items oi JOIN products p ON oi.product_id = p.product_id WHERE oi.order_id = 10;
```

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

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

Extra options

order_item_id	order_id	product_id	quantity	unit_price	product_name
1	10	1	2	50.00	first aid kits

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

Query results operations

Print Copy to clipboard Export Display chart Create view

The screenshot shows the phpMyAdmin interface with a sidebar containing a tree view of database tables. The main area displays a query result for order items where the order ID is 10. The result set contains one row with the following data:

order_item_id	order_id	product_id	quantity	unit_price	product_name
1	10	1	2	50.00	first aid kits

49. Update order status:

```
UPDATE orders SET order_status = 'Shipped' , shipped_date = NOW()
WHERE order_id = 10;
```

The screenshot shows the phpMyAdmin interface for MySQL. The left sidebar lists various database tables: New, database, New, cart_items, categories, coupons, coupon_usages, orders, order_items, payments, products, product_attributes, product_attribute_values, product_categories, product_images, product_reviews, product_views, refunds, shipping_methods, users, user_addresses, and wishlists. The main area is titled 'Table: orders' and contains a SQL query: `UPDATE orders SET order_status = 'Shipped', shipped_date = NOW() WHERE order_id = 10;`. A message at the top indicates '1 row affected. (Query took 0.0010 seconds.)'. Below the message are three buttons: 'Edit inline', 'Edit', and 'Create PHP code'.

50.Cancel an order:

UPDATE orders SET order_status = 'Cancelled' WHERE order_id = 10

This screenshot is identical to the one above, showing the phpMyAdmin interface for MySQL. The left sidebar lists the same set of tables. The main area shows the same SQL query: `UPDATE orders SET order_status = 'Cancelled', shipped_date = NOW() WHERE order_id = 10;`. The message '1 row affected. (Query took 0.0010 seconds.)' is displayed, along with the 'Edit inline', 'Edit', and 'Create PHP code' buttons.

51.Get orders by status (e.g., 'Pending'):

SELECT * FROM orders WHERE order_status = 'Pending'

The screenshot shows the phpMyAdmin interface for a MySQL database named 'database'. The 'orders' table is selected. A single row is visible in the results grid:

order_id	user_id	order_date	total_amount	order_status	shipping_address_id	billing_address_id	tracking_number	shipped_date	delivered_date
1	1	2025-07-01 16:43:27	150.00	Pending	1	NULL	NULL	NULL	NULL

52. Get recent orders (last 7 days):

```
SELECT * FROM orders WHERE order_date >= DATE_SUB(NOW(), INTERVAL 7 DAY);
```

The screenshot shows the phpMyAdmin interface for a MySQL database named 'database'. The 'orders' table is selected. Two rows are visible in the results grid:

order_id	user_id	order_date	total_amount	order_status	shipping_address_id	billing_address_id	tracking_number	shipped_date	delivered_date
1	1	2025-07-01 16:43:27	150.00	Pending	1	NULL	NULL	NULL	NULL
10	1	2025-07-01 16:43:27	150.00	Cancelled	1	NULL	NULL	2025-07-01 16:53:16	NULL

53. Get total sales for a specific period:

```
SELECT SUM(total_amount) FROM orders WHERE order_date BETWEEN '2025-01-01' AND '2025-01-31' AND order_status NOT IN ('Cancelled', 'Refunded')
```

The screenshot shows the phpMyAdmin interface with the following details:

- Server:** MySQL:3306
- Database:** database
- Table:** orders
- Query:** SELECT SUM(total_amount) FROM orders WHERE order_date BETWEEN '2025-01-01' AND '2025-01-31' AND order_status NOT IN ('Cancelled', 'Refunded');
- Result:** SUM(total_amount) NULL
- Operations:** Print, Copy to clipboard, Export, Display chart, Create view

54. Apply a discount to an order (e.g., by updating total_amount or adding a discount record):

```
UPDATE orders SET total_amount = total_amount * 0.90 WHERE
order_id = 10; -- 10% discount
```

The screenshot shows the phpMyAdmin interface with the following details:

- Server:** MySQL:3306
- Database:** database
- Table:** orders
- Query:** UPDATE orders SET total_amount = total_amount * 0.90 WHERE order_id = 10;
- Result:** 1 row affected. (Query took 0.0008 seconds)
- Message:** MySQL returned an empty result set (i.e. zero rows). (Query took 0.0003 seconds)
- Operations:** Edit inline, Edit, Create PHP code

55. Record payment for an order:

```
INSERT INTO payments (order_id, payment_method, amount,
transaction_id, payment_status, payment_date)
VALUES (10, 'Credit Card', 150.00, 'TXN12345', 'Completed',
NOW());
```

The screenshot shows the phpMyAdmin interface for a MySQL database named 'database'. The left sidebar lists various tables: New, database, cart_items, categories, coupons, coupon_usages, orders, order_items, payments, products, product_attributes, product_attribute_values, product_categories, product_images, product_reviews, product_views, refunds, shipping_methods, users, and user_addresses. The main area shows a query result for an 'INSERT INTO' statement into the 'payments' table:

```
INSERT INTO payments (order_id, payment_method, amount, transaction_id, payment_status, payment_date) VALUES (10, 'Credit Card', 150.00, 'TXN12345', 'Completed', NOW());
```

A green message bar at the top indicates "1 row inserted. Inserted row id: 1 (Query took 0.0064 seconds.)".

56. Get payment details for an order:

```
SELECT * FROM payments WHERE order_id = 10
```

The screenshot shows the phpMyAdmin interface for the same MySQL database. The main area displays the results of the previous SQL query:

```
SELECT * FROM payments WHERE order_id = 10;
```

The results table shows one row:

payment_id	order_id	payment_method	amount	transaction_id	payment_status	payment_date
1	10	Credit Card	150.00	TXN12345	Completed	2025-07-01 17:00:14

57. Process a refund:

```
INSERT INTO refunds (order_id, refund_amount, refund_date, reason)
```

```
VALUES (10, 50.00, NOW(), 'Customer changed mind');
```

```
UPDATE orders SET order_status = 'Refunded' WHERE order_id = 10
```

```

Server: MySQL3306 > Database: database > Table: payments
Show query box
1 row inserted.
Inserted row id: 1 (Query took 0.0046 seconds.)
INSERT INTO refunds (order_id, refund_amount, refund_date, reason) VALUES (10, 50.00, NOW(), 'Customer changed mind');

[Edit inline] [Edit] [Create PHP code]

1 row affected. (Query took 0.0006 seconds.)
UPDATE orders SET order_status = 'Refunded' WHERE order_id = 10;

[Edit inline] [Edit] [Create PHP code]

```

58. Reviews & Ratings

Submit a product review:

INSERT INTO product_reviews (product_id, user_id, rating,

review_text, review_date)

VALUES (1, 1, 5, 'Excellent product, highly recommend!', NOW())

```

Server: MySQL3306 > Database: database > Table: payments
Show query box
1 row inserted.
Inserted row id: 1 (Query took 0.0070 seconds.)
INSERT INTO product_reviews (product_id, user_id, rating, review_text, review_date) VALUES (1, 1, 5, 'Excellent product, highly recommend!', NOW());

[Edit inline] [Edit] [Create PHP code]

```

59. Get all reviews for a product:

SELECT pr.*, u.username

FROM product_reviews pr

JOIN users u ON pr.user_id = u.user_id

WHERE pr.product_id = 1

ORDER BY pr.review_date DESC

Showing rows 0 - 0 (1 total, Query took 0.0154 seconds.) (review_date: 2025-07-01 17:03:31.. - 2025-07-01 17:03:31..)

```
SELECT pr.*, u.username FROM product_reviews pr JOIN users u ON pr.user_id = u.user_id WHERE pr.product_id = 1 ORDER BY pr.review_date DESC;
```

review_id	product_id	user_id	rating	review_text	review_date	is_approved	username
1	1	1	5	Excellent product, highly recommend!	2025-07-01 17:03:31	0	john_doe

60. Get average rating for a product:

```
SELECT AVG(rating) AS average_rating FROM product_reviews WHERE
product_id = 1;
```

Showing rows 0 - 0 (1 total, Query took 0.0051 seconds.)

```
SELECT AVG(rating) AS average_rating FROM product_reviews WHERE product_id = 1;
```

average_rating
5.0000

61. Get total number of reviews for a product:

```
SELECT COUNT(*) AS total_reviews FROM product_reviews WHERE
product_id = 1;
```

```

Server: MySQL:3306 > Database: database > Table: product_reviews
Show query box
Your SQL query has been executed successfully.

SELECT COUNT(*) AS total_reviews FROM product_reviews WHERE product_id = 1;

total_reviews
1

```

62. Get reviews by a specific user:

SELECT * FROM product_reviews WHERE user_id = 1 ORDER BY review_date DESC

review_id	product_id	user_id	rating	review_text	review_date	is_approved
1	1	1	5	Excellent product, highly recommend!	2025-07-01 17:03:31	0

63. Update a product review:

UPDATE product_reviews SET review_text = 'Still great, but price increased.', rating = 4 WHERE review_id = 1;

The screenshot shows the phpMyAdmin interface for a MySQL database. The left sidebar lists various tables: New, database, cart_items, categories, coupons, coupon_usages, orders, order_items, payments, products, product_attributes, product_attribute_values, product_categories, product_images, product_reviews, product_views, refunds, shipping_methods, users, user_addresses, and wishlists. The main panel shows the results of an UPDATE query on the 'product_reviews' table:

```
✓ 1 row affected. (Query took 0.0019 seconds.)  
UPDATE product_reviews SET review_text = 'Still great, but price increased.', rating = 4 WHERE review_id = 1;
```

Below the query results are links for [Edit inline], [Edit], and [Create PHP code].

64.Delete a product review (admin/user):

DELETE FROM product_reviews WHERE review_id = 1

The screenshot shows the phpMyAdmin interface for a MySQL database. The left sidebar lists the same set of tables as the previous screenshot. The main panel shows the results of a DELETE query on the 'product_reviews' table:

```
✓ 1 row deleted. (Query took 0.0010 seconds.)  
DELETE FROM product_reviews WHERE review_id = 1;
```

Below the query results are links for [Edit inline], [Edit], and [Create PHP code].

65.Discounts & Coupons

Create a new coupon:

```
INSERT INTO coupons (coupon_code, discount_type, discount_value,  
valid_from, valid_until, usage_limit, min_order_amount)  
VALUES ('SAVE10', 'percentage', 10.00, NOW(), '2025-12-31', 100,  
50.00);
```

The screenshot shows the phpMyAdmin interface for a MySQL database. The left sidebar lists various tables: New, database, cart_items, categories, coupons, coupon_usages, orders, order_items, payments, products, product_attributes, product_attribute_values, product_categories, product_images, product_reviews, product_views, refunds, shipping_methods, users, and user_addresses. The main area shows a query result for an 'INSERT INTO coupons' operation. The message says '1 row inserted.' and 'Inserted row id: 5 (Query took 0.0007 seconds.)'. The SQL query is:

```
INSERT INTO coupons (coupon_code, discount_type, discount_value, valid_from, valid_until, usage_limit, min_order_amount) VALUES ('SAVE10', 'percentage', 10.00, NOW(), '2025-12-31', 100, 50.00);
```

Buttons for 'Edit inline', 'Edit', and 'Create PHP code' are visible.

66. Get coupon details by code:

```
SELECT * FROM coupons WHERE coupon_code = 'SAVE10'
```

The screenshot shows the phpMyAdmin interface for a MySQL database. The left sidebar lists various tables. The main area displays the results of a 'SELECT * FROM coupons WHERE coupon_code = 'SAVE10'' query. The results table shows one row:

coupon_id	coupon_code	discount_type	discount_value	valid_from	valid_until	usage_limit	times_used	min_order_amount	is_active	created_at
5	SAVE10	percentage	10.00	2025-07-01	2025-12-31	100	0	50.00	1	2025-07-01 17:14:09

Buttons for 'Edit', 'Copy', 'Delete', 'Show all', 'Number of rows: 25', 'Filter rows: Search this table', 'Extra options', 'With selected:', 'Edit', 'Copy', 'Delete', 'Export', 'Show all', 'Number of rows: 25', 'Filter rows: Search this table', 'Query results operations', 'Print', 'Copy to clipboard', 'Export', 'Display chart', and 'Create view' are visible.

67. Update coupon details:

```
UPDATE coupons SET discount_value = 15.00 WHERE coupon_code = 'SAVE10'
```

The screenshot shows the phpMyAdmin interface for a MySQL database named 'database'. The left sidebar lists tables such as New, database, cart_items, categories, coupons, coupon_usages, orders, order_items, payments, products, product_attributes, product_attribute_values, product_categories, and product_images. The main panel shows the 'coupons' table. A query has been run:

```
UPDATE coupons SET discount_value = 15.00 WHERE coupon_code = 'SAVE10';
```

The result message indicates 0 rows affected.

68.delete a coupon:

```
DELETE FROM coupons WHERE coupon_id = 1
```

The screenshot shows the phpMyAdmin interface for a MySQL database named 'database'. The left sidebar lists tables including coupons. The main panel shows the 'coupons' table. A query has been run:

```
DELETE FROM coupons WHERE coupon_id = 1;
```

The result message indicates 0 rows deleted.

69.Record coupon usage:

```
INSERT INTO coupon_usages (coupon_id, user_id, order_id,
```

```
usage_date)
```

```
VALUES (1, 1, 10, NOW());
```

```
UPDATE coupons SET times_used = times_used + 1 WHERE coupon_id = 1
```

```

Server: MySQL:3306 > Database: database > Table: coupon_usages
Browse Structure SQL Search Insert Export Import Privileges Operations Triggers

Show query box

✓ 1 row inserted.
Inserted row id: 3 (Query took 0.0008 seconds.)
INSERT INTO coupon_usages (coupon_id, user_id, order_id, usage_date) VALUES (1, 1, 10, NOW());

[Edit inline] [Edit] [Create PHP code]

✓ 0 rows affected. (Query took 0.0005 seconds.)
UPDATE coupons SET times_used = times_used + 1 WHERE coupon_id = 1;

[Edit inline] [Edit] [Create PHP code]

```

70. Check if a coupon is valid (date, usage limit, min order amount):

```

SELECT * FROM coupons
WHERE coupon_code = 'SAVE10'
AND valid_from <= NOW()
AND valid_until >= NOW()
AND times_used < usage_limit
AND 120.00 >= min_order_amount; -- Assuming 120.00 is the current
order total

```

```

Server: MySQL:3306 > Database: database > Table: coupons
Browse Structure SQL Search Insert Export Import Privileges Operations Triggers

Show query box

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0006 seconds.)

SELECT * FROM coupons WHERE coupon_code = 'SAVE10' AND valid_from <= NOW() AND valid_until >= NOW() AND times_used < usage_limit AND 120.00 >= min_order_amount;

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

coupon_id coupon_code discount_type discount_value valid_from valid_until usage_limit times_used min_order_amount is_active created_at

Query results operations
Create view

```

71. Shipping & Inventory

Get all shipping methods:

`SELECT * FROM shipping_methods`

The screenshot shows the phpMyAdmin interface for a MySQL database named 'database'. The left sidebar lists various tables, including 'cart_items', 'categories', 'coupons', 'coupon_usages', 'orders', 'order_items', 'payments', 'products', 'product_attributes', 'product_attribute_values', 'product_categories', 'product_images', 'product_reviews', 'product_views', 'refunds', 'shipping_methods', 'users', 'user_addresses', and 'wishlist'. The main area displays the results of the query `SELECT * FROM shipping_methods;`. The results table has columns: method_id, method_name, cost, estimated_delivery_days, and is_active. Two rows are shown:

	method_id	method_name	cost	estimated_delivery_days	is_active
1	1	method-1	3499.00	7	1
2	2	method-2	2500.00	6	1

72.Add a new shipping method:

`INSERT INTO shipping_methods (method_name, cost,`

`estimated_delivery_days)`

`VALUES ('Standard Shipping', 5.99, 5);`

The screenshot shows the phpMyAdmin interface for a MySQL database named 'database'. The left sidebar lists various tables, including 'cart_items', 'categories', 'coupons', 'coupon_usages', 'orders', 'order_items', 'payments', 'products', 'product_attributes', 'product_attribute_values', 'product_categories', 'product_images', 'product_reviews', 'product_views', 'refunds', 'shipping_methods', 'users', 'user_addresses', and 'wishlist'. The main area displays the results of the query `INSERT INTO shipping_methods (method_name, cost, estimated_delivery_days) VALUES ('Standard Shipping', 5.99, 5);`. A message at the top indicates '1 row inserted.' and 'Inserted row id: 3 (Query took 0.00007 seconds.)'. The results table shows the newly added row:

	method_id	method_name	cost	estimated_delivery_days	is_active
1	1	method-1	3499.00	7	1
2	2	method-2	2500.00	6	1
3	3	Standard Shipping	5.99	5	1

73.Update shipping method cost:

`UPDATE shipping_methods SET cost = 7.50 WHERE method_id = 1`

The screenshot shows the phpMyAdmin interface for MySQL. The left sidebar lists various database tables under the 'database' section. The main query window displays the following SQL command:

```
UPDATE shipping_methods SET cost = 7.50 WHERE method_id = 1;
```

The status bar at the bottom indicates "1 row affected. (Query took 0.0005 seconds.)".

74. Get product stock quantity:

```
SELECT stock_quantity FROM products WHERE product_id = 1
```

The screenshot shows the phpMyAdmin interface for MySQL. The left sidebar lists various database tables under the 'database' section. The main query window displays the following SQL command:

```
SELECT stock_quantity FROM products WHERE product_id = 1;
```

The status bar at the bottom indicates "Showing rows 0 - 0 (1 total, Query took 0.0016 seconds.)".

75. Decrease product stock after an order:

```
UPDATE products SET stock_quantity = stock_quantity - 2 WHERE  
product_id = 1
```

The screenshot shows the phpMyAdmin interface for MySQL. The left sidebar lists various database tables under the 'database' section. The main query window displays a successful UPDATE operation on the 'products' table:

```
✓ 1 row affected. (Query took 0.0008 seconds.)  
UPDATE products SET stock_quantity = stock_quantity + 2 WHERE product_id = 1;
```

76. Increase product stock (e.g., for returns or new inventory):

UPDATE products SET stock_quantity = stock_quantity + 5 WHERE

product_id = 1

The screenshot shows the phpMyAdmin interface for MySQL. The left sidebar lists various database tables under the 'database' section. The main query window displays a successful UPDATE operation on the 'products' table:

```
✓ 1 row affected. (Query took 0.0079 seconds.)  
UPDATE products SET stock_quantity = stock_quantity + 5 WHERE product_id = 1;
```

77. Get products that are out of stock:

SELECT * FROM products WHERE stock_quantity = 0

Showing rows 0 - 0 (1 total, Query took 0.0017 seconds.)

```
SELECT * FROM products WHERE stock_quantity = 0;
```

product_id	product_name	description	regular_price	sale_price	stock_quantity	sku	weight	dimensions	created_at	updated_at	is_active
3	White Casual Wears	a variety of comfortable and relaxed clothing opt...	500.00	250.00	0	TS-BLM-A101	16.0 17.0	M 40.0 26.0 L 42.0 28.0 ...	2025-06-30 10:35:09	2025-07-01 18:28:50	1

78.Admin & Reporting

Get total number of users:

```
SELECT COUNT(*) FROM users
```

Your SQL query has been executed successfully.

```
SELECT COUNT(*) FROM users;
```

COUNT(*)

2

79.Get total number of products:

```
SELECT COUNT(*) FROM products;
```

The screenshot shows the phpMyAdmin interface for a MySQL database named 'database'. The current table is 'products'. The left sidebar lists various tables: database, New, cart_items, categories, coupons, coupon_usages, order_items, orders, payments, products, product_attributes, product_attribute_values, product_categories, product_images, product_reviews, product_views, refunds, shipping_methods, users, user_addresses, and wishlists. The main query results show the output of the SQL query: 'SELECT COUNT(*) FROM products;'. The result is COUNT(*) = 4. Below the results, there are options to Print, Copy to clipboard, Export, Display chart, and Create view.

```
Server: MySQL:3306 > Database: database > Table: products
Browse Structure SQL Search Insert Export Import Privileges Operations Triggers
Show query box
Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.
Your SQL query has been executed successfully.
SELECT COUNT(*) FROM products;
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Extra options
COUNT(*) 4
Query results operations
Print Copy to clipboard Export Display chart Create view
```

80. Get total number of orders:

```
SELECT COUNT(*) FROM orders
```

The screenshot shows the phpMyAdmin interface for a MySQL database named 'database'. The current table is 'orders'. The left sidebar lists various tables: database, New, cart_items, categories, coupons, coupon_usages, order_items, orders, payments, products, product_attributes, product_attribute_values, product_categories, product_images, product_reviews, product_views, refunds, shipping_methods, users, and user_addresses. The main query results show the output of the SQL query: 'SELECT COUNT(*) FROM orders;'. The result is COUNT(*) = 2. Below the results, there are options to Print, Copy to clipboard, Export, Display chart, and Create view.

```
Server: MySQL:3306 > Database: database > Table: orders
Browse Structure SQL Search Insert Export Import Privileges Operations Triggers
Show query box
Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.
Your SQL query has been executed successfully.
SELECT COUNT(*) FROM orders;
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Extra options
COUNT(*) 2
Query results operations
Print Copy to clipboard Export Display chart Create view
```

81. Get revenue by month:

```
SELECT DATE_FORMAT(order_date, '%Y-%m') AS sales_month,
SUM(total_amount) AS monthly_revenue
FROM orders
WHERE order_status NOT IN ('Cancelled', 'Refunded')
GROUP BY sales_month
ORDER BY sales_month
```

The screenshot shows the phpMyAdmin interface for a MySQL database named 'database'. The current table is 'orders'. The query executed is:

```
SELECT DATE_FORMAT(order_date, '%Y-%m') AS sales_month, SUM(total_amount) AS monthly_revenue FROM orders WHERE order_status NOT IN ('Cancelled', 'Refunded') GROUP BY sales_month ORDER BY sales_month;
```

The results show one row:

sales_month	monthly_revenue
2025-07	285.00

82. Get top 10 best-selling products (by quantity):

```
SELECT p.product_name, SUM(oi.quantity) AS total_quantity_sold
FROM order_items oi
JOIN products p ON oi.product_id = p.product_id
GROUP BY p.product_name
ORDER BY total_quantity_sold DESC
LIMIT 10
```

The screenshot shows the phpMyAdmin interface for a MySQL database named 'database'. The current table is 'p'. The query executed is:

```
SELECT p.product_name, SUM(oi.quantity) AS total_quantity_sold FROM order_items oi JOIN products p ON oi.product_id = p.product_id GROUP BY p.product_name ORDER BY total_quantity_sold DESC LIMIT 10;
```

The results show one row:

product_name	total_quantity_sold
first aid kits	2

83. Get top 10 customers (by total spend):

```
SELECT u.username, SUM(o.total_amount) AS total_spent
FROM orders o
JOIN users u ON o.user_id = u.user_id
WHERE o.order_status NOT IN ('Cancelled', 'Refunded')
```

```

GROUP BY u.username
ORDER BY total_spent DESC
LIMIT 10;

```

The screenshot shows the MySQL Workbench interface. The left sidebar lists various database tables. The main area displays a query results window with the following content:

```

SELECT u.username, SUM(o.total_amount) AS total_spent FROM orders o JOIN users u ON o.user_id = u.user_id WHERE o.order_status NOT IN ('Cancelled', 'Refunded') GROUP BY u.username ORDER BY total_spent DESC LIMIT 10;

```

The results table shows one row:

username	total_spent
john_doe	285.00

Below the table are several operation buttons: Print, Copy to clipboard, Export, Display chart, and Create view.

84. Get average order value:

```

SELECT AVG(total_amount) FROM orders WHERE order_status NOT IN
('Cancelled', 'Refunded')

```

The screenshot shows the phpMyAdmin interface. The left sidebar lists various database tables. The main area displays a query results window with the following content:

```

SELECT AVG(total_amount) FROM orders WHERE order_status NOT IN ('Cancelled', 'Refunded');

```

The results table shows one row:

AVG(total_amount)
142.50000

Below the table are several operation buttons: Show all, Number of rows: 25, Filter rows: Search this table, Show all, Number of rows: 25, Filter rows: Search this table, Print, Copy to clipboard, Export, Display chart, and Create view.

85. Get orders with pending shipments:

```

SELECT * FROM orders WHERE order_status = 'Processed'; -- Or
'Pending Shipping'

```

Server: MySQL:3306 » Database: database » Table: orders

Browse Structure SQL Search Insert Export Import Privileges Operations Triggers

Show query box

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0006 seconds.)

SELECT * FROM orders WHERE order_status = 'Processed';

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

order_id user_id order_date total_amount order_status shipping_address_id billing_address_id tracking_number shipped_date delivered

Query results operations

Create view

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0002 seconds.)

-- Or 'Pending Shipping';

[Edit inline] [Edit] [Create PHP code]

86. Get customers who haven't placed an order:

```
SELECT u.*  
FROM users u  
LEFT JOIN orders o ON u.user_id = o.user_id  
WHERE o.order_id IS NULL;
```

Server: MySQL:3306 » Database: database » Table: users

Browse Structure SQL Search Insert Export Import Privileges Operations Triggers

Show query box

Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.

Showing rows 0 - 0 (1 total, Query took 0.0020 seconds.)

SELECT u.* FROM users u LEFT JOIN orders o ON u.user_id = o.user_id WHERE o.order_id IS NULL;

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

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

user_id	username	email	password_hash	first_name	last_name	phone_number	registration_date	last_login	is_active	role	created_at
2	john	sowmi@gmail.com	hashed_password	NULL	NULL	NULL	2025-07-01 12:12:52	NULL	1	customer	2025-07-01

Query results operations

Print Copy to clipboard Export Display chart Create view

87. Get products with no reviews:

```
SELECT p.*  
FROM products p  
LEFT JOIN product_reviews pr ON p.product_id = pr.product_id  
WHERE pr.review_id IS NULL
```

Showing rows 0 - 3 (4 total, Query took 0.0061 seconds)

```
SELECT p.* FROM products p LEFT JOIN product_reviews pr ON p.product_id = pr.product_id WHERE pr.review_id IS NULL;
```

product_id	product_name	description	regular_price	sale_price	stock_quantity	sku	weight	dimensions	created_at	updated_at	is_active
1	first aid kits	Class A and Class B	2500.00	1500.00	10	ANSI A+	650.00	225 x 235 x 95mm	2025-06-30 10:35:09	2025-07-01 18:26:03	1
3	White Casual Wears	a variety of comfortable and relaxed clothing opt...	500.00	250.00	0	TS-BLM-A101	156.00	S 38.0 24.0 16.0 M 40.0 26.0 L 42.0 28.0 ...	2025-06-30 10:35:09	2025-07-01 18:28:50	1
101	New Gadget	A cool new device.	99.99	NULL	100	NULL	NULL	NULL	2025-06-30 11:00:48	2025-07-01 16:39:46	1
102	New Gadget	A cool new device.	99.99	NULL	100	NULL	NULL	NULL	2025-07-01 11:13:43	2025-07-01 16:39:33	1

88. Get product categories with their product count:

```
SELECT c.category_name, COUNT(pc.product_id) AS product_count
FROM categories c
LEFT JOIN product_categories pc ON c.category_id = pc.category_id
GROUP BY c.category_name
ORDER BY product_count DESC
```

Showing rows 0 - 4 (5 total, Query took 0.0105 seconds)

```
SELECT c.category_name, COUNT(pc.product_id) AS product_count FROM categories c LEFT JOIN product_categories pc ON c.category_id = pc.category_id GROUP BY c.category_name ORDER BY product_count DESC;
```

category_name	product_count
Electronics	0
Health Care	0
Fashion	0
Furniture	0
Electricals	0

89. Get daily sales trend:

```
SELECT DATE(order_date) AS sales_day, SUM(total_amount) AS daily_revenue
FROM orders
WHERE order_status NOT IN ('Cancelled', 'Refunded')
GROUP BY sales_day
```

ORDER BY sales_day

The screenshot shows the phpMyAdmin interface for a MySQL database named 'material_db'. The left sidebar lists various tables: database, New, cart_items, categories, coupons, coupon_usages, orders, order_items, payments, products, product_attributes, product_attribute_values, product_categories, product_images, product_reviews, product_views, refunds, shipping_methods, users, user_addresses, wishlists, and material_db. The main panel displays a query result for the 'orders' table. The query is:

```
SELECT DATE(order_date) AS sales_day, SUM(total_amount) AS daily_revenue FROM orders WHERE order_status NOT IN ('Cancelled', 'Refunded') GROUP BY sales_day ORDER BY sales_day;
```

The results show two rows:

sales_day	daily_revenue
2025-07-01	135.00
2025-07-02	150.00

Below the results, there are buttons for Print, Copy to clipboard, Export, Display chart, and Create view.

90. Get products that are frequently purchased together (simple example):

```
SELECT oi1.product_id AS product1_id, oi2.product_id AS product2_id, COUNT(*) AS co_occurrence_count
FROM order_items oi1
JOIN order_items oi2 ON oi1.order_id = oi2.order_id AND
oi1.product_id < oi2.product_id
GROUP BY product1_id, product2_id
ORDER BY co_occurrence_count DESC
LIMIT 10
```

The screenshot shows the phpMyAdmin interface for a MySQL database named 'material_db'. The left sidebar lists various tables: database, New, cart_items, categories, coupons, coupon_usages, orders, order_items, payments, products, product_attributes, product_attribute_values, product_categories, product_images, product_reviews, product_views, refunds, shipping_methods, users, user_addresses, wishlists, and material_db. The main panel displays a query result for the 'order_items' table. The query is:

```
SELECT oi1.product_id AS product1_id, oi2.product_id AS product2_id, COUNT(*) AS co_occurrence_count FROM order_items oi1 JOIN order_items oi2 ON oi1.order_id = oi2.order_id AND oi1.product_id < oi2.product_id GROUP BY product1_id, product2_id ORDER BY co_occurrence_count DESC LIMIT 10;
```

The results show one row:

product1_id	product2_id	co_occurrence_count
1	5	1

Below the results, there are buttons for Print, Copy to clipboard, Export, Display chart, and Create view.

91. Miscellaneous/Advanced

Calculate lifetime value (LTV) for each customer:

```

SELECT u.user_id, u.username, SUM(o.total_amount) AS
lifetime_value
FROM users u
JOIN orders o ON u.user_id = o.user_id
WHERE o.order_status NOT IN ('Cancelled', 'Refunded')
GROUP BY u.user_id, u.username
ORDER BY lifetime_value DESC;

```

Showing rows 0 - 0 (1 total). Query took 0.0025 seconds.

```

SELECT u.user_id, u.username, SUM(o.total_amount) AS lifetime_value
FROM users u
JOIN orders o ON u.user_id = o.user_id
WHERE o.order_status NOT IN ('Cancelled', 'Refunded')
GROUP BY u.user_id, u.username
ORDER BY lifetime_value DESC;

```

user_id	username	lifetime_value
1	john_doe	285.00

92. Get user session activity (if you have a sessions table):

```

SELECT * FROM user_sessions WHERE user_id = 1 ORDER BY
last_activity DESC

```

Showing rows 0 - 0 (1 total). Query took 0.0048 seconds.) [last_activity: [BLOB - 0 B]... - [BLOB - 0 B]...]

```

SELECT * FROM user_sessions WHERE user_id = 1 ORDER BY last_activity DESC;

```

	id	user_id	date_created	status	sessions	last_activity
	1	1	2025-07-01 19:04:32	1	morning	

93. Record product view:

```

INSERT INTO product_views (product_id, user_id, view_time)

```

`VALUES (1, 1, NOW())`

The screenshot shows the phpMyAdmin interface. In the left sidebar, there's a tree view of database tables under 'New'. The main area shows a query result: '1 row inserted.' Below it is the SQL query: `INSERT INTO product_views (product_id, user_id, view_time) VALUES (1, 1, NOW());`. The top navigation bar includes tabs for 'Browse', 'Structure', 'SQL', 'Search', 'Insert', 'Export', 'Import', 'Privileges', 'Operations', and 'Triggers'.

94. Get products recently viewed by a user:

```
SELECT DISTINCT p.*  
FROM product_views pv  
JOIN products p ON pv.product_id = p.product_id  
WHERE pv.user_id = 1  
ORDER BY pv.view_time DESC  
LIMIT 10;
```

The screenshot shows the phpMyAdmin interface after running the provided SQL query. The results table displays 10 rows of products recently viewed by user ID 1. The columns include product_id, product_name, description, regular_price, sale_price, stock_quantity, sku, weight, dimensions, created_at, updated_at, and is_active. The first row is highlighted.

product_id	product_name	description	regular_price	sale_price	stock_quantity	sku	weight	dimensions	created_at	updated_at	is_active
1	first aid kits	Class A and Class B	2500.00	1500.00	10	ANSI A+	650.00	225 x 235 x 95mm	2025-06-30 10:35:09	2025-07-01 18:26:03	1

95. Manage email subscriptions (if you have a newsletter table):

```
INSERT INTO newsletter_subscribers (email, subscribe_date) VALUES  
('user@example.com', NOW())
```

The screenshot shows the phpMyAdmin interface for MySQL. The left sidebar lists various database tables. The main area shows a query result: "1 row inserted. Inserted row id: 2 (Query took 0.0007 seconds.)". Below it is the SQL query: "INSERT INTO newsletter_subscribers (email, subscribe_date) VALUES ('user@example.com', NOW());". A warning message at the bottom says: "Warning: #1264 Out of range value for column 'subscribe_date' at row 1".

96.Unsubscribe from newsletter:

```
DELETE FROM newsletter_subscribers WHERE email =
'user@example.com'
```

The screenshot shows the phpMyAdmin interface for MySQL. The left sidebar lists various database tables. The main area shows a query result: "1 row deleted. (Query took 0.0006 seconds.)". Below it is the SQL query: "DELETE FROM newsletter_subscribers WHERE email = 'user@example.com';".

97.Get all products with their associated categories:

```
SELECT p.product_name, GROUP_CONCAT(c.category_name SEPARATOR ',
') AS categories
FROM products p
LEFT JOIN product_categories pc ON p.product_id = pc.product_id
LEFT JOIN categories c ON pc.category_id = c.category_id
GROUP BY p.product_id;
```

The screenshot shows the phpMyAdmin interface for a MySQL database. The left sidebar lists various tables: New, database, cart_items, categories, coupons, coupon_usages, newsletter_subscribers, orders, order_items, payments, products, product_attributes, product_attribute_values, product_categories, product_images, product_reviews, product_views, refunds, shipping_methods, users, and user_addresses. The main area displays a query result for the 'products' table. The query is:

```
SELECT p.product_name, GROUP_CONCAT(c.category_name SEPARATOR ', ') AS categories FROM products p LEFT JOIN product_categories pc ON p.product_id = pc.product_id LEFT JOIN categories c ON pc.category_id = c.category_id GROUP BY p.product_id;
```

The results show four rows:

product_name	categories
first aid kits	NULL
White Casual Wears	NULL
New Gadget	NULL
New Gadget	NULL

98. Find duplicate product names (for data cleaning):

```
SELECT product_name, COUNT(*)
```

```
FROM products
```

```
GROUP BY product_name
```

```
HAVING COUNT(*) > 1
```

The screenshot shows the phpMyAdmin interface after executing the query. The message bar at the top says "Your SQL query has been executed successfully." The query shown is:

```
SELECT product_name, COUNT(*) FROM products GROUP BY product_name HAVING COUNT(*) > 1;
```

The results table shows one row:

product_name	COUNT(*)
New Gadget	2

99. Get users who have abandoned their cart:

```
SELECT u.*
```

```
FROM users u
```

```
JOIN cart_items ci ON u.user_id = ci.user_id
```

```
LEFT JOIN orders o ON u.user_id = o.user_id AND o.order_date >
```

```
ci.added_at -- Assuming 'added_at' for cart item
```

WHERE o.order_id IS NULL

GROUP BY u.user_id;

The screenshot shows the phpMyAdmin interface for a MySQL database. The left sidebar lists various tables: New, database, New, cart_items, categories, coupons, coupon_usages, newsletter_subscribers, orders, order_items, payments, products, product_attributes, product_attribute_values, product_categories, product_images, product_reviews, product_views, refunds, shipping_methods, users, and user_addresses. The main panel displays a query result for the 'users' table. The SQL query is:

```
SELECT u.* FROM users u JOIN cart_items ci ON u.user_id = ci.user_id LEFT JOIN orders o ON u.user_id = o.user_id AND o.order_date > ci.added_at -- Assuming 'added_at' for cart item WHERE o.order_id IS NULL GROUP BY u.user_id;
```

The results show one row for user_id 2, John, with email sowni@gmail.com. The 'order_id' column is listed as 'NULL'.

100. Archive old orders (moving them to an archive table for performance):

CREATE TABLE archived_orders AS

SELECT * FROM orders

WHERE order_date < DATE_SUB(NOW(), INTERVAL 2 YEAR);

The screenshot shows the phpMyAdmin interface for a MySQL database. The left sidebar lists the same set of tables as the previous screenshot. The main panel displays a query result for the 'orders' table. The SQL query is:

```
SELECT * FROM orders WHERE order_date < DATE_SUB(NOW(), INTERVAL 2 YEAR);
```

The results show zero rows returned, indicating no old orders found.