execute-query

mysqli::execute_query

mysqli_execute_query

Prepares, binds parameters, and executes SQL statement

ENTITY

ENTITY

public

mysqli_result

bool

mysqli::execute_query

string

query

array

null

params

ENTITY

ENTITY

mysqli_result

bool

mysqli_execute_query

mysqli

mysql

string

query

array

null

params

ENTITY

Prepares the SQL query, binds parameters, and executes it. The mysqli::execute_query method is a shortcut for mysqli::prepare, mysqli_stmt::bind_param, mysqli_stmt::execute, and mysqli_stmt::get_result.

The statement template can contain zero or more question mark (`?`) parameter markers—also called placeholders. The parameter values must be provided as an ENTITY using `params` parameter.

A prepared statement is created under the hood but itENTITYs never exposed outside of the function. ItENTITYs impossible to access properties of the statement as one would do with the `mysqli_stmt` object. Due to this limitation, the status information is copied to the `mysqli` object and is available using its methods, e.g. `mysqli_affected_rows` or `mysqli_error`.

> [!NOTE] In the case where a statement is passed to `mysqli_execute_query` that is longer than `max_allowed_packet` of the server, the returned error codes are different depending on the operating system. The behavior is as follows:
>
> - On Linux returns an error code of 1153. The error message means “got a packet bigger than `max_allowed_packet` bytes”.
> - On Windows returns an error code 2006. This error message means “server has gone away”.

ENTITY

`query`
The query, as a string. It must consist of a single SQL statement.

The SQL statement may contain zero or more parameter markers represented by question mark (`?`) characters at the appropriate positions.

> [!NOTE] The markers are legal only in certain places in SQL statements. For example, they are permitted in the `VALUES()` list of an `INSERT` statement (to specify column values for a row), or in a comparison with a column in a `WHERE` clause to specify a comparison value. However, they are not permitted for identifiers (such as table or column names).

`params`
An optional list ENTITY with as many elements as there are bound parameters in the SQL statement being executed. Each value is treated as a ENTITY.

ENTITY

Returns ENTITY on failure. For successful queries which produce a result set, such as `SELECT, SHOW, DESCRIBE` or `EXPLAIN`, returns a `mysqli_result` object. For other successful queries, returns ENTITY.

ENTITY

mysqli::execute_query example

ENTITY

```
<?php

mysqli_report(MYSQLI_REPORT_ERROR | MYSQLI_REPORT_STRICT);
$mysqli = new mysqli('localhost', 'my_user', 'my_password', 'world');

$query = 'SELECT Name, District FROM City WHERE CountryCode=? ORDER BY Name LIMIT 5';
$result = $mysqli->execute_query($query, ['DEU']);
foreach ($result as $row) {
    printf("%s (%s)\n", $row["Name"], $row["District"]);
}
```

ENTITY

```
<?php
```

```
mysqli_report(MYSQLI_REPORT_ERROR | MYSQLI_REPORT_STRICT);
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

$query = 'SELECT Name, District FROM City WHERE CountryCode=? ORDER BY Name LIMIT 5';
$result = mysqli_execute_query($link, $query, ['DEU']);
foreach ($result as $row) {
    printf("%s (%s)\n", $row["Name"], $row["District"]);
}
```

ENTITY

```
Aachen (Nordrhein-Westfalen)
Augsburg (Baijeri)
Bergisch Gladbach (Nordrhein-Westfalen)
Berlin (Berliini)
Bielefeld (Nordrhein-Westfalen)
```

ENTITY

`mysqli_prepare`, `mysqli_stmt_execute`, `mysqli_stmt_bind_param`, `mysqli_stmt_get_result`

□□□ #2
□ □□□ □□□ 2026-10-31 17:56:01 UTC
□ □□□ □□□ 2026-10-31 17:56:02 UTC