Select Data From a MySQL Database

The SELECT statement is used to select data from one or more tables:

```
SELECT column_name(s) FROM table_name
```

or we can use the * character to select ALL columns from a table:

```
SELECT * FROM table_name
```

To learn more about SQL, please visit our SQL tutorial.

Select Data With MySQLi

The following example selects the id, firstname and lastname columns from the MyGuests table and displays it on the page:

Example (MySQLi Object-oriented)

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

$sql = "SELECT id, firstname, lastname FROM MyGuests";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
  // output data of each row
  while($row = $result->fetch_assoc()) {
    echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " . $row["lastname"]. "<br>";
  }
} else {
  echo "0 results";
}
$conn->close();
?>
```

First, we set up an SQL query that selects the id, firstname and lastname columns from the MyGuests table. The next line of code runs the query and puts the resulting data into a variable called $result.

Then, the function num_rows() checks if there are more than zero rows returned.

If there are more than zero rows returned, the function `fetch_assoc()` puts all the results into an associative array that we can loop through. The `while()` loop loops through the result set and outputs the data from the id, firstname and lastname columns.

The following example shows the same as the example above, in the MySQLi procedural way:

Example (MySQLi Procedural)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
  die("Connection failed: " . mysqli_connect_error());
}

$sql = "SELECT id, firstname, lastname FROM MyGuests";
$result = mysqli_query($conn, $sql);

if (mysqli_num_rows($result) > 0) {
  // output data of each row
  while($row = mysqli_fetch_assoc($result)) {
    echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " . $row["lastname"]. "<br>";
  }
} else {
  echo "0 results";
}

mysqli_close($conn);
?>
```

You can also put the result in an HTML table:

Example (MySQLi Object-oriented)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

$sql = "SELECT id, firstname, lastname FROM MyGuests";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
  echo "<table><tr><th>ID</th><th>Name</th></tr>";
```

```
  // output data of each row
  while($row = $result->fetch_assoc()) {
    echo "<tr><td>".$row["id"]."</td><td>".$row["firstname"]."
".$row["lastname"]."</td></tr>";
  }
  echo "</table>";
} else {
  echo "0 results";
}
$conn->close();
?>
```

Select Data With PDO (+ Prepared Statements)

The following example uses prepared statements.

It selects the id, firstname and lastname columns from the MyGuests table and displays it in an HTML table:

Example (PDO)

```
<?php
echo "<table style='border: solid 1px black;'>";
echo "<tr><th>Id</th><th>Firstname</th><th>Lastname</th></tr>";

class TableRows extends RecursiveIteratorIterator {
  function __construct($it) {
    parent::__construct($it, self::LEAVES_ONLY);
  }

  function current() {
    return "<td style='width:150px;border:1px solid black;'>" . parent::current(). "</td>";
  }

  function beginChildren() {
    echo "<tr>";
  }

  function endChildren() {
    echo "</tr>" . "\n";
  }
}

$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDBPDO";

try {
  $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);
  $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  $stmt = $conn->prepare("SELECT id, firstname, lastname FROM MyGuests");
  $stmt->execute();

  // set the resulting array to associative
  $result = $stmt->setFetchMode(PDO::FETCH_ASSOC);
  foreach(new TableRows(new RecursiveArrayIterator($stmt->fetchAll())) as $k=>$v) {
    echo $v;
```

```
  }
} catch(PDOException $e) {
  echo "Error: " . $e->getMessage();
}
$conn = null;
echo "</table>";
?>
```

Select and Filter Data From a MySQL Database

The WHERE clause is used to filter records.

The WHERE clause is used to extract only those records that fulfill a specified condition.

```
SELECT column_name(s) FROM table_name WHERE column_name operator value
```

To learn more about SQL, please visit our SQL tutorial.

Select and Filter Data With MySQLi

The following example selects the id, firstname and lastname columns from the MyGuests table where the lastname is "Doe", and displays it on the page:

Example (MySQLi Object-oriented)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

$sql = "SELECT id, firstname, lastname FROM MyGuests WHERE lastname='Doe'";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
  // output data of each row
  while($row = $result->fetch_assoc()) {
    echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " . $row["lastname"]. "<br>";
  }
} else {
  echo "0 results";
}
$conn->close();
?>
```

First, we set up the SQL query that selects the id, firstname and lastname columns from the MyGuests table where the lastname is "Doe". The next line of code runs the query and puts the resulting data into a variable called $result.

Then, the `function num_rows()` checks if there are more than zero rows returned.

If there are more than zero rows returned, the function `fetch_assoc()` puts all the results into an associative array that we can loop through. The `while()` loop loops through the result set and outputs the data from the id, firstname and lastname columns.

The following example shows the same as the example above, in the MySQLi procedural way:

Example (MySQLi Procedural)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
  die("Connection failed: " . mysqli_connect_error());
}

$sql = "SELECT id, firstname, lastname FROM MyGuests WHERE lastname='Doe'";
$result = mysqli_query($conn, $sql);

if (mysqli_num_rows($result) > 0) {
  // output data of each row
  while($row = mysqli_fetch_assoc($result)) {
    echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " . $row["lastname"]. "<br>";
  }
} else {
  echo "0 results";
}

mysqli_close($conn);
?>
```

Example (MySQLi Object-oriented)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

$sql = "SELECT id, firstname, lastname FROM MyGuests WHERE lastname='Doe'";
$result = $conn->query($sql);
```

```php
if ($result->num_rows > 0) {
  echo "<table><tr><th>ID</th><th>Name</th></tr>";
  // output data of each row
  while($row = $result->fetch_assoc()) {
    echo "<tr><td>".$row["id"]."</td><td>".$row["firstname"]."
".$row["lastname"]."</td></tr>";
  }
  echo "</table>";
} else {
  echo "0 results";
}
$conn->close();
?>
```

Select Data With PDO (+ Prepared Statements)

The following example uses prepared statements.

It selects the id, firstname and lastname columns from the MyGuests table where the lastname is "Doe", and displays it in an HTML table:

Example (PDO)

```php
<?php
echo "<table style='border: solid 1px black;'>";
echo "<tr><th>Id</th><th>Firstname</th><th>Lastname</th></tr>";

class TableRows extends RecursiveIteratorIterator {
  function __construct($it) {
    parent::__construct($it, self::LEAVES_ONLY);
  }

  function current() {
    return "<td style='width:150px;border:1px solid black;'>" . parent::current(). "</td>";
  }

  function beginChildren() {
    echo "<tr>";
  }

  function endChildren() {
    echo "</tr>" . "\n";
  }
}

$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDBPDO";

try {
  $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);
  $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  $stmt = $conn->prepare("SELECT id, firstname, lastname FROM MyGuests WHERE lastname='Doe'");
  $stmt->execute();

  // set the resulting array to associative
```

```
  $result = $stmt->setFetchMode(PDO::FETCH_ASSOC);
  foreach(new TableRows(new RecursiveArrayIterator($stmt->fetchAll())) as $k=>$v) {
    echo $v;
  }
}
catch(PDOException $e) {
  echo "Error: " . $e->getMessage();
}
$conn = null;
echo "</table>";
?>
```

Select and Order Data From a MySQL Database

The ORDER BY clause is used to sort the result-set in ascending or descending order.

The ORDER BY clause sorts the records in ascending order by default. To sort the records in descending order, use the DESC keyword.

```
SELECT column_name(s) FROM table_name ORDER BY column_name(s) ASC|DESC
```

To learn more about SQL, please visit our SQL tutorial.

Select and Order Data With MySQLi

The following example selects the id, firstname and lastname columns from the MyGuests table. The records will be ordered by the lastname column:

Example (MySQLi Object-oriented)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

$sql = "SELECT id, firstname, lastname FROM MyGuests ORDER BY lastname";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
  // output data of each row
  while($row = $result->fetch_assoc()) {
    echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " . $row["lastname"]. "<br>";
  }
} else {
  echo "0 results";
```

```
}
$conn->close();
?>
```

First, we set up the SQL query that selects the id, firstname and lastname columns from the MyGuests table. The records will be ordered by the lastname column. The next line of code runs the query and puts the resulting data into a variable called $result.

Then, the `function num_rows()` checks if there are more than zero rows returned.

If there are more than zero rows returned, the function `fetch_assoc()` puts all the results into an associative array that we can loop through. The `while()` loop loops through the result set and outputs the data from the id, firstname and lastname columns.

The following example shows the same as the example above, in the MySQLi procedural way:

Example (MySQLi Procedural)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
  die("Connection failed: " . mysqli_connect_error());
}

$sql = "SELECT id, firstname, lastname FROM MyGuests ORDER BY lastname";
$result = mysqli_query($conn, $sql);

if (mysqli_num_rows($result) > 0) {
  // output data of each row
  while($row = mysqli_fetch_assoc($result)) {
    echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " . $row["lastname"]. "<br>";
  }
} else {
  echo "0 results";
}

mysqli_close($conn);
?>
```

Example (MySQLi Object-oriented)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
```

```
}

$sql = "SELECT id, firstname, lastname FROM MyGuests ORDER BY lastname";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
  echo "<table><tr><th>ID</th><th>Name</th></tr>";
  // output data of each row
  while($row = $result->fetch_assoc()) {
    echo "<tr><td>".$row["id"]."</td><td>".$row["firstname"]."
".$row["lastname"]."</td></tr>";
  }
  echo "</table>";
} else {
  echo "0 results";
}
$conn->close();
?>
```

Run example »

Select Data With PDO (+ Prepared Statements)

The following example uses prepared statements.

Here we select the id, firstname and lastname columns from the MyGuests table. The records will be ordered by the lastname column, and it will be displayed in an HTML table:

Example (PDO)

```
<?php
echo "<table style='border: solid 1px black;'>";
echo "<tr><th>Id</th><th>Firstname</th><th>Lastname</th></tr>";

class TableRows extends RecursiveIteratorIterator {
  function __construct($it) {
    parent::__construct($it, self::LEAVES_ONLY);
  }

  function current() {
    return "<td style='width:150px;border:1px solid black;'>" . parent::current(). "</td>";
  }

  function beginChildren() {
    echo "<tr>";
  }

  function endChildren() {
    echo "</tr>" . "\n";
  }
}

$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDBPDO";
```

```php
try {
  $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);
  $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  $stmt = $conn->prepare("SELECT id, firstname, lastname FROM MyGuests ORDER BY lastname");
  $stmt->execute();

  // set the resulting array to associative
  $result = $stmt->setFetchMode(PDO::FETCH_ASSOC);
  foreach(new TableRows(new RecursiveArrayIterator($stmt->fetchAll())) as $k=>$v) {
    echo $v;
  }
} catch(PDOException $e) {
  echo "Error: " . $e->getMessage();
}
$conn = null;
echo "</table>";
?>
```

Delete Data From a MySQL Table Using MySQLi and PDO

The DELETE statement is used to delete records from a table:

```
DELETE FROM table_name
WHERE some_column = some_value
```

Notice the WHERE clause in the DELETE syntax: The WHERE clause specifies which record or records that should be deleted. If you omit the WHERE clause, all records will be deleted!

To learn more about SQL, please visit our SQL tutorial.

Let's look at the "MyGuests" table:

id	firstname	lastname	email	reg_date
1	John	Doe	john@example.com	2014-10-22 14:26:15
2	Mary	Moe	mary@example.com	2014-10-23 10:22:30
3	Julie	Dooley	julie@example.com	2014-10-26 10:48:23

The following examples delete the record with id=3 in the "MyGuests" table:

Example (MySQLi Object-oriented)

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

// sql to delete a record
$sql = "DELETE FROM MyGuests WHERE id=3";

if ($conn->query($sql) === TRUE) {
  echo "Record deleted successfully";
} else {
  echo "Error deleting record: " . $conn->error;
}

$conn->close();
?>
```

Example (MySQLi Procedural)

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
  die("Connection failed: " . mysqli_connect_error());
}

// sql to delete a record
$sql = "DELETE FROM MyGuests WHERE id=3";

if (mysqli_query($conn, $sql)) {
  echo "Record deleted successfully";
} else {
  echo "Error deleting record: " . mysqli_error($conn);
}

mysqli_close($conn);
?>
```

Example (PDO)

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDBPDO";
```

```php
try {
  $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);
  // set the PDO error mode to exception
  $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

  // sql to delete a record
  $sql = "DELETE FROM MyGuests WHERE id=3";

  // use exec() because no results are returned
  $conn->exec($sql);
  echo "Record deleted successfully";
} catch(PDOException $e) {
  echo $sql . "<br>" . $e->getMessage();
}

$conn = null;
?>
```

After the record is deleted, the table will look like this:

id	firstname	lastname	email	reg_date
1	John	Doe	john@example.com	2014-10-22 14:26:15
2	Mary	Moe	mary@example.com	2014-10-23 10:22:30

Update Data In a MySQL Table Using MySQLi and PDO

The UPDATE statement is used to update existing records in a table:

```
UPDATE table_name
SET column1=value, column2=value2,...
WHERE some_column=some_value
```

Notice the WHERE clause in the UPDATE syntax: The WHERE clause specifies which record or records that should be updated. If you omit the WHERE clause, all records will be updated!

To learn more about SQL, please visit our SQL tutorial.

Let's look at the "MyGuests" table:

id	firstname	lastname	email	reg_date

1	John	Doe	john@example.com	2014-10-22 14:26:15
2	Mary	Moe	mary@example.com	2014-10-23 10:22:30

The following examples update the record with id=2 in the "MyGuests" table:

Example (MySQLi Object-oriented)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

$sql = "UPDATE MyGuests SET lastname='Doe' WHERE id=2";

if ($conn->query($sql) === TRUE) {
  echo "Record updated successfully";
} else {
  echo "Error updating record: " . $conn->error;
}

$conn->close();
?>
```

Example (MySQLi Procedural)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
  die("Connection failed: " . mysqli_connect_error());
}

$sql = "UPDATE MyGuests SET lastname='Doe' WHERE id=2";

if (mysqli_query($conn, $sql)) {
  echo "Record updated successfully";
} else {
  echo "Error updating record: " . mysqli_error($conn);
}
```

```
mysqli_close($conn);
?>
```

Example (PDO)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDBPDO";

try {
  $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);
  // set the PDO error mode to exception
  $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

  $sql = "UPDATE MyGuests SET lastname='Doe' WHERE id=2";

  // Prepare statement
  $stmt = $conn->prepare($sql);

  // execute the query
  $stmt->execute();

  // echo a message to say the UPDATE succeeded
  echo $stmt->rowCount() . " records UPDATED successfully";
} catch(PDOException $e) {
  echo $sql . "<br>" . $e->getMessage();
}

$conn = null;
?>
```

After the record is updated, the table will look like this:

id	firstname	lastname	email	reg_date
1	John	Doe	john@example.com	2014-10-22 14:26:15
2	Mary	Doe	mary@example.com	2014-10-23 10:22:30

Limit Data Selections From a MySQL Database

MySQL provides a LIMIT clause that is used to specify the number of records to return.

The LIMIT clause makes it easy to code multi page results or pagination with SQL, and is very useful on large tables. Returning a large number of records can impact on performance.

Assume we wish to select all records from 1 - 30 (inclusive) from a table called "Orders". The SQL query would then look like this:

```
$sql = "SELECT * FROM Orders LIMIT 30";
```

When the SQL query above is run, it will return the first 30 records.

What if we want to select records 16 - 25 (inclusive)?

Mysql also provides a way to handle this: by using OFFSET.

The SQL query below says "return only 10 records, start on record 16 (OFFSET 15)":

```
$sql = "SELECT * FROM Orders LIMIT 10 OFFSET 15";
```

You could also use a shorter syntax to achieve the same result:

```
$sql = "SELECT * FROM Orders LIMIT 15, 10";
```

Notice that the numbers are reversed when you use a comma.