MySQL PHP Tutorial - Select & Display Data From MySQL Tables

clevertechie.com/php/100/mysql-php-tutorial-select-display-data-from-mysql-tables

PHP | 22 Feb, 2017 | Clever Techie
Google +1 17 0

MySQL PHP Tutorial - Select & Display Data From MySQL Tables

Welcome back! In this tutorial we’re going to be selecting the MySQL data records from a database and displaying them using HTML and CSS. We’re going to be displaying 20 movies ordered randomly from the database along with all of their information. The full source code is below but I’ll be walking through portions of it beneath that. In this particular tutorial I won’t be going over the HTML or CSS but there will be another video on those. If you don’t have the database or any of the IMDb data we’re using, go back to the previous videos and grab the source code.

```
<?php

$host = 'localhost';

$user = 'root';

$pass = 'mypass123';

$db = 'movies';

$mysqli = new mysqli($host,$user,$pass,$db);

$result = $mysqli->query

("SELECT * FROM movies WHERE year BETWEEN 2000 AND 2016 ORDER BY rand()
LIMIT 20")

or die($mysqli->error);

?><divclass='main-container'><?php

while ($movie = $result->fetch_assoc()): ?>

<divclass='movie-container'>

<divclass='header'>

<h1><?= $movie['title'] ?></h1>

<spanclass='year'>( <?= $movie['year'] ?> )</span>

</div>
```

```
<divclass='content'>

<divclass='left-column'>

<imgwidth='<?php 67*1.3 ?>'height='<?= 98*1.3 ?>'src='<?= $movie['image_url'] ?>'>

<divid='ratings'>

<divclass='imdb'><?= $movie['imdb_rating'] ? $movie['imdb_rating'] : '' ?></div>

<divclass='metascore'><?= $movie['metascore'] ? $movie['metascore'] : '' ?></div>

</div>

</div>

<divclass='right-column'>

<spanclass='content blue'>

<?= $movie['certificate']; ?>

</span>

<?php

echo $movie['certificate'] ? ' |' : '';

?>

<spanclass='content blue'>

<?= $movie['runtime'] .' min'; ?>

</span>

<?php

$result2 = $mysqli->query

("SELECT genres_id FROM movies_genres WHERE movies_id={$movie['id']}") or

die($mysqli->error);

$genres = $result2->fetch_all();

$genres = array_column($genres, 0);

for ($i = 0; $i < count($genres);$i++)

{

$genre = $mysqli->query("SELECT name from genres where id = '{$genres[$i]}'")-
>fetch_assoc();

echo $i == 0 ? ' | ' : '';
echo "<span class='content blue'>".$genre['name']."</span>";

echo $genres[$i] != end($genres) ? ', ' : '';

}

?>

<divclass='content description'><?= $movie['description'] ?></div>

<?php

$result3 = $mysqli->query

("SELECT directors_id FROM movies_directors WHERE movies_id={$movie['id']}") or

die($mysqli->error);

$directors = $result3->fetch_all();

$directors = array_column($directors, 0);

$result4 = $mysqli->query

("SELECT stars_id FROM movies_stars WHERE movies_id={$movie['id']}") or

die($mysqli->error);

$stars = $result4->fetch_all();

$stars = array_column($stars, 0);

?>

<div>

<?php

for ($i = 0; $i < count($directors);$i++)

{

$director = $mysqli->query

("SELECT name from directors where id = '{$directors[$i]}'")->fetch_assoc();

$s = count($directors) > 1 ? 's' : '';

echo $i == 0 ? "<span class='content yellow'>Director$s: </span>" : '';

echo "<span class='content text'>".$director['name']."</span>";

if ($directors[$i] != end($directors)){

echo ', ';
}

else {

if (count($stars) > 0)

{

echo ' | ';

}

}

}

?>

<?php

for ($i = 0; $i < count($stars);$i++)

{

$star = $mysqli->query("SELECT name from stars where id = '{$stars[$i]}'")-
>fetch_assoc();

$s = count($stars) > 1 ? 's' : '';

echo $i == 0 ? "<span class='content yellow'>Star$s: </span>" : '';

echo "<span>".$star['name']."</span>";

echo $stars[$i] != end($stars) ? ', ' : '';

}

?>

</div>

<divclass='bottom'>

<?php

if ($movie['votes']) {

echo "<span class='content yellow'>Votes: </span>".number_format($movie['votes']);

echo $movie['gross'] ? ' | ' : '';

}

?>

<spanclass='content green'>
```

<?= $movie['gross'] ? "<span class='content yellow'>Gross: </span>$".

number_format($movie['gross']) : '' ?>

</span>

</div>

</div>

</div>

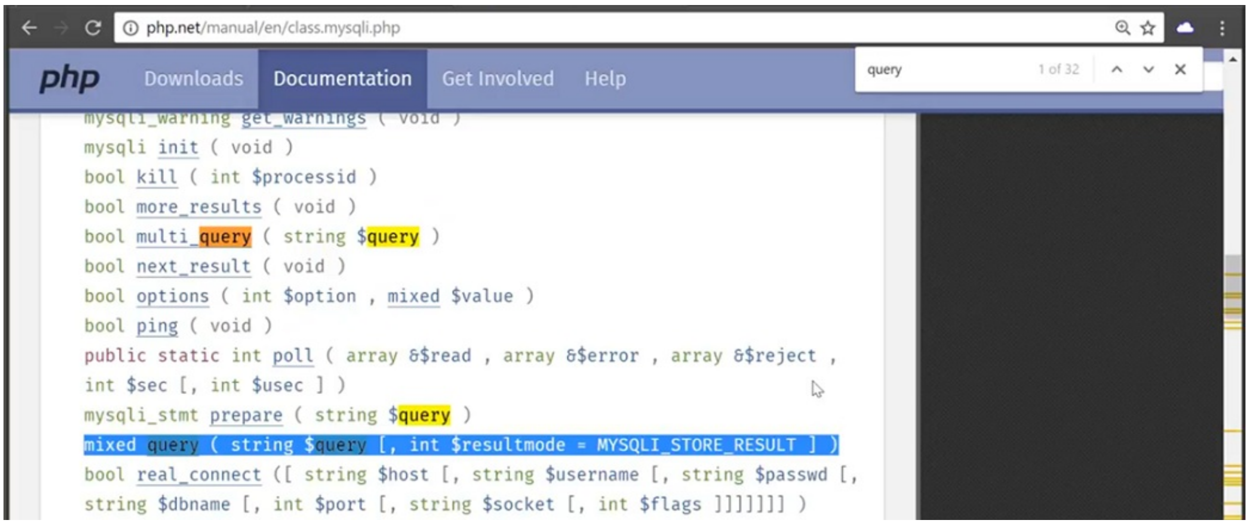
</div>

</div>

<?php endwhile; ?>

MySQLi and MySQLi Result Objects

First, include the style.css and connect to the database as we've done before. Next we'll create a mysqli object by calling the class with keyword new. Whenever we have an object variable we can call methods on it, and the object can have properties. Some of the methods that can be called on a mysqli object are shown below, with the method "query" highlighted.



```
mysqli_warning get_warnings ( void )
mysqli init ( void )
bool kill ( int $processid )
bool more_results ( void )
bool multi_query ( string $query )
bool next_result ( void )
bool options ( int $option , mixed $value )
bool ping ( void )
public static int poll ( array &$read , array &$error , array &$reject ,
int $sec [, int $usec ] )
mysqli_stmt prepare ( string $query )
mixed query ( string $query [, int $resultmode = MYSQLI_STORE_RESULT ] )
bool real_connect ([ string $host [, string $username [, string $passwd [,
string $dbname [, int $port [, string $socket [, int $flags ]]]]]] )
```

We're going to call the method "query" on our mysqli object, and the result of that is going to be a mysqli result object. The query selects 20 random movies from the database where the year is from 2000 to 2006.

$host = 'localhost';

$user = 'root';

$pass = 'mypass123';

$db = 'movies';

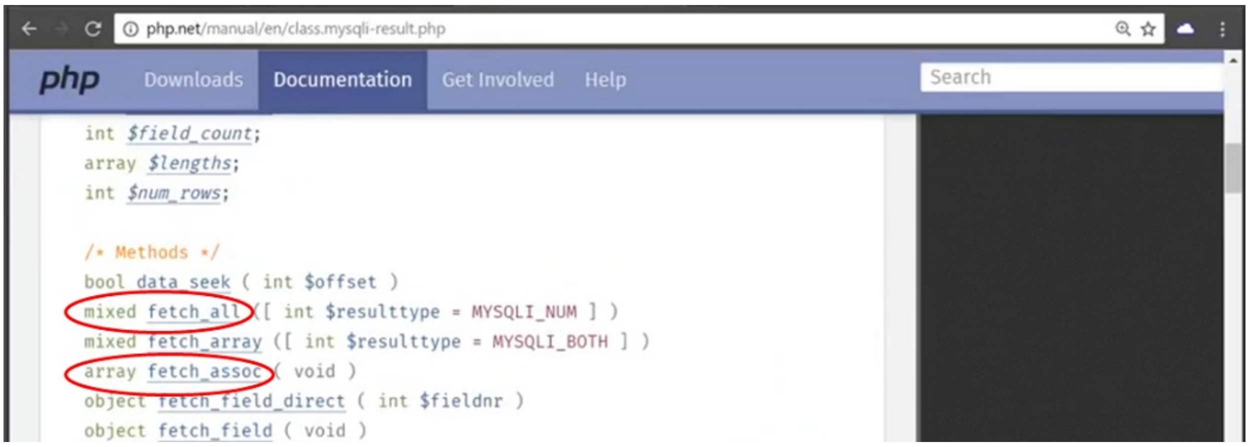
$mysqli = new mysqli($host,$user,$pass,$db);

$result = $mysqli->query

("SELECT * FROM movies WHERE year BETWEEN 2000 AND 2016 ORDER BY rand() LIMIT 20")

or die($mysqli->error);

The result object has its own properties and methods, and two of the methods we'll be using on the result object are fetch_all and fetch_assoc, shown below. When we call fetch_assoc it will return a row's column data and then when it is called again it gets a new row. Since it automatically fetches the next row every time it is called until no row is returned, we will put it directly in our while loop, looping through the result until we've gone through all 20 movies.



```
int $field_count;
array $lengths;
int $num_rows;

/* Methods */
bool data_seek ( int $offset )
mixed fetch_all ([ int $resulttype = MYSQLI_NUM ] )
mixed fetch_array ([ int $resulttype = MYSQLI_BOTH ] )
array fetch_assoc ( void )
object fetch_field_direct ( int $fieldnr )
object fetch_field ( void )
```

while ($movie = $result->fetch_assoc()): ?>

Displaying Formatted Data

As fetch_assoc pulls each movie during the while loop, they are stored as an associative array in $movie. Using HTML we print out the movie title inside a header and the year inside parentheses. To print out the image of the movie, the src attribute is set to the image URL. I already know that the image is 67-by-98 but it's a little small, so I multiply each of those dimensions by 1.3.

<divclass='movie-container'>

<divclass='header'>

<h1><?= $movie['title'] ?></h1>

<spanclass='year'>(<?= $movie['year'] ?>)</span>

</div>

<divclass='content'>

<divclass='left-column'>

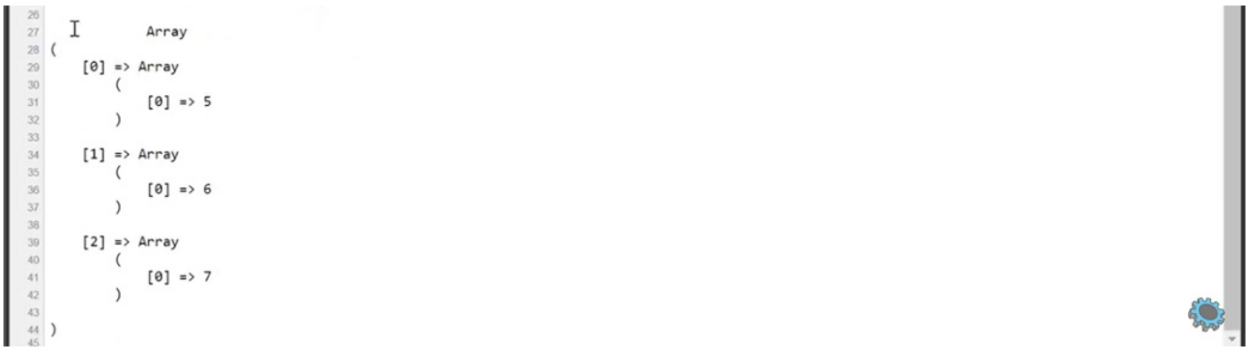
<imgwidth='<?php 67*1.3 ?>'height='<?= 98*1.3 ?>'src='<?= $movie['image_url'] ?>'>

Next we want to print out the IMDb rating and metascore if they exist. We use the ternary operator to print them if they exist and print nothing if they do not. In a similar vein, we want to print the certificate if it exists, but if it does exist we also want to print a pipe afterwards to come between it and the runtime. The runtime is printed next, with the abbreviation for minutes added afterwards.

```
<divid='ratings'>

<divclass='imdb'><?= $movie['imdb_rating'] ? $movie['imdb_rating'] : '' ?></div>

<divclass='metascore'><?= $movie['metascore'] ? $movie['metascore'] : '' ?></div>

</div>

</div>

<divclass='right-column'>

<spanclass='content blue'>

<?= $movie['certificate']; ?>

</span>

<?php

echo $movie['certificate'] ? ' |' : '';

?>

<spanclass='content blue'>

<?= $movie['runtime'] .' min'; ?>

</span>
```

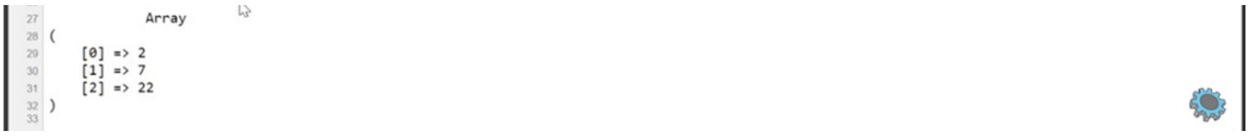
Selecting Data From Relational Tables

This is where things get a little trickier. We're going to be selecting data from the relational tables, beginning with the genres. We use query again to select all the movie genre IDs associated with the current movie ID. The query returns a result object and we can use fetch_all to get all the genre IDs in a nicely formatted numeric array. Unfortunately, if we look at what is returned we see that it is a multidimensional array as shown below.



```
        Array
(
    [0] => Array
        (
            [0] => 5
        )

    [1] => Array
        (
            [0] => 6
        )

    [2] => Array
        (
            [0] => 7
        )

)
```

To clean it up we can use the array_column function. The code shown below takes in the array and deletes the 0 key to result in a nicely formatted array.



```
Array
(
    [0] => 2
    [1] => 7
    [2] => 22
)
```

$result2 = $mysqli->query

("SELECT genres_id FROM movies_genres WHERE movies_id={$movie['id']}") or

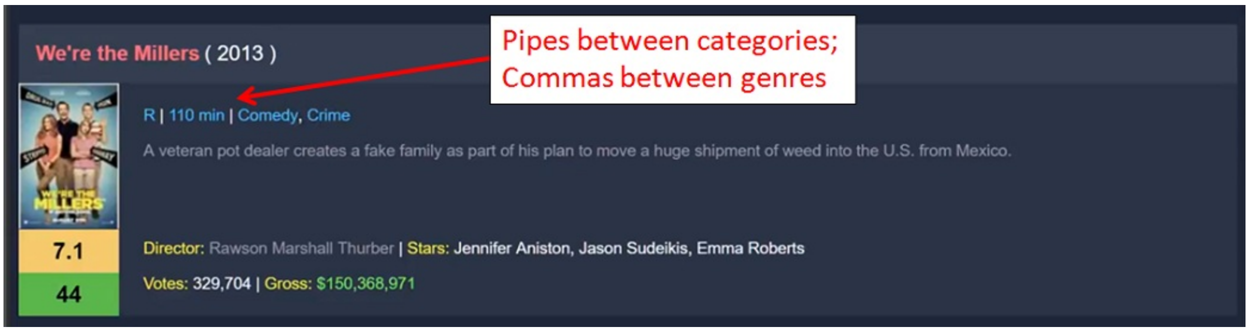
die($mysqli->error);

$genres = $result2->fetch_all();

$genres = array_column($genres, 0);

With the array of genre IDs ready, we create a for loop to loop through them and query the genre name from the genres table. Since this is going to be a result object we can immediately call a method on it instead of storing it to a new variable first, so we'll call fetch_assoc on it to immediately return an associative array.

We also want to print a pipe before the first genre, so we check to see whether the counter is zero before printing the genre name. If the counter is zero it prints a pipe before printing the genre name. We also check to see whether this is the final genre or not so we know whether to print a comma afterwards.



We're the Millers (2013)

R | 110 min | Comedy, Crime

A veteran pot dealer creates a fake family as part of his plan to move a huge shipment of weed into the U.S. from Mexico.

Director: Rawson Marshall Thurber | Stars: Jennifer Aniston, Jason Sudeikis, Emma Roberts

Votes: 329,704 | Gross: $150,368,971

7.1

44

for ($i = 0; $i < count($genres);$i++)

{

$genre = $mysqli->query("SELECT name from genres where id = '{$genres[$i]}'")->fetch_assoc();

echo $i == 0 ? ' | ' : '';

echo "<span class='content blue'>".$genre['name']."</span>";

echo $genres[$i] != end($genres) ? ', ' : '';

}

The movie description is very simple to display with no real tricks to it. That line of code is given below.

```
<divclass='content description'><?= $movie['description'] ?></div>
```

The directors and stars follow the same principle as the genres: we call query to get the list of director or star IDs, then we use fetch_all and array_column to format them into our arrays. The code block below follows this procedure to get and store the list of director and star IDs.

```
$result3 = $mysqli->query
("SELECT directors_id FROM movies_directors WHERE movies_id={$movie['id']}") or
die($mysqli->error);
$directors = $result3->fetch_all();
$directors = array_column($directors, 0);
$result4 = $mysqli->query
("SELECT stars_id FROM movies_stars WHERE movies_id={$movie['id']}") or
die($mysqli->error);
$stars = $result4->fetch_all();
$stars = array_column($stars, 0);
```

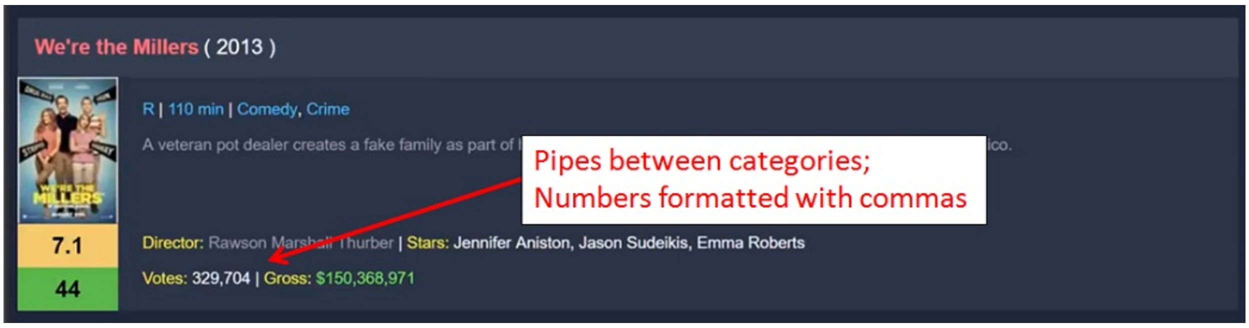
With our arrays of director and star IDs, we can create a loop to go through the directors. We loop through, using query and fetch_assoc to get the director names, and if there is more than one director we store the letter s to a variable $s. Later we will append this to the word “director” to be printed. Again, a comma is printed after every director name except the last one.

Within the director loop we also check to see whether there are any stars listed for this movie. If we have printed director names and there will be stars next to them, we want to print a pipe in between.

```
for ($i = 0; $i < count($directors);$i++)
{
$director = $mysqli->query
("SELECT name from directors where id = '{$directors[$i]}'")->fetch_assoc();
$s = count($directors) > 1 ? 's' : '';
echo $i == 0 ? "<span class='content yellow'>Director$s: </span>" : '';
echo "<span class='content text'>".$director['name']."</span>";
if ($directors[$i] != end($directors)){
echo ', ';
}

else {

if (count($stars) > 0)

{

echo ' | ';

}

}

}
```

The process of looping through the stars is very similar so I won’t dwell on it. The code for this portion is included in the source code above.

The final categories to display are the votes and gross. For each of these we need to check if they exist and then format them appropriately. We use number_format so they will be displayed with commas. Within the if statement displaying the votes we also check to see whether gross is empty so we can display a pipe between the two if both are present.



We're the Millers (2013)

R | 110 min | Comedy, Crime

A veteran pot dealer creates a fake family as part of [illegible] ico.

Director: Rawson Marshall Thurber | Stars: Jennifer Aniston, Jason Sudeikis, Emma Roberts

Votes: 329,704 | Gross: $150,368,971

```
<divclass='bottom'>

<?php

if ($movie['votes']) {

echo "<span class='content yellow'>Votes: </span>".number_format($movie['votes']);

echo $movie['gross'] ? ' | ' : '';

}

?>

<spanclass='content green'>

<?= $movie['gross'] ? "<span class='content yellow'>Gross: </span>$".

number_format($movie['gross']) : '' ?>

</span>
```

```
</div>
```

I hope this tutorial was interesting for you. There will be a video in the future on how to create CSS for this. You can play around and keep refreshing the page to see the different random movies, and you can also play around with the SQL statement to change the parameters. If you found this tutorial useful be sure to like, share, and subscribe to Clever Techie!