

Inserting Data Into MySQL Database Relational Tables Using PHP

 clevertechie.com/php/99/inserting-data-into-mysql-database-relational-tables-using-php

PHP | 20 Feb, 2017 | Clever Techie

Google +0 27 0

Inserting Data Into MySQL Database Relationship Tables Using PHP

Welcome back! In this tutorial we're going to learn how to insert data into a MySQL database which has many-to-many relational tables. We'll insert the information we've scraped from IMDb into our MySQL database so that later we can display it in a nicely formatted way. The full source code is below but skip down below that and I'll walk through each portion and how it works.

```
<?php

include('scrape_imdb.php');

$data = scrape_imdb(1999, 2017, 1, 2);

$host = 'localhost';

$user = 'root';

$pass = 'mypass123';

$db = 'movies';

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

for ($i = 0; $i < count($data); $i++ ) {

    $description = $mysqli->escape_string($data[$i]['description']);

    $title = $mysqli->escape_string($data[$i]['title']);

    $votes = intval($data[$i]['votes']);

    $gross = intval(str_replace(',','',$data[$i]['gross']));

    $sql = "INSERT IGNORE INTO movies (title,year,image_url,certificate,runtime,
    imdb_rating,metascore,description,votes,gross)

    VALUES ('$title','{$data[$i]['year']}','{$data[$i]['image']}','
    '{$data[$i]['certificate']}','{$data[$i]['runtime']}','{$data[$i]['imdb_rating']},
```

```

'{$data[$i]['metascore']}', '$description', '$votes', '$gross')";

$mysqli->query($sql) or die($mysqli->error);

$movies_id = $mysqli->insert_id;

if ( !$movies_id ) {

continue;

}

$directors = explode(",", $data[$i]['directors']);

$stars = explode(",", $data[$i]['stars']);

$genres = explode(",", $data[$i]['genres']);

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

{

$director = $mysqli->escape_string(trim($directors[$c]));

$sql = "INSERT IGNORE INTO directors (name) VALUES ('$director')";

$mysqli->query($sql) or die($mysqli->error);

$directors_id = $mysqli->insert_id;

if ( $directors_id ){

$sql = "INSERT INTO movies_directors (movies_id, directors_id) "

. "VALUES ('$movies_id', '$directors_id')";

$mysqli->query($sql);

}

else {

$sql = "SELECT id FROM directors WHERE name='$director'";

$result = $mysqli->query($sql) or die($mysqli->error);

$row = $result->fetch_assoc();

$directors_id = $row['id'];

$sql = "INSERT INTO movies_directors (movies_id, directors_id) "

. "VALUES ('$movies_id', '$directors_id')";

$mysqli->query($sql);

```

```

}

}

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

{

$star = $mysqli->escape_string(trim($stars[$c]));

$sql = "INSERT IGNORE INTO stars (name) VALUES ('$star')";

$mysqli->query($sql) or die($mysqli->error);

$stars_id = $mysqli->insert_id;

if ( $stars_id ){

$sql = "INSERT INTO movies_stars (movies_id, stars_id) "

. "VALUES ('$movies_id','$stars_id')";

$mysqli->query($sql);

}

else {

$sql = "SELECT id FROM stars WHERE name='$star'";

$result = $mysqli->query($sql) or die($mysqli->error);

$row = $result->fetch_assoc();

$stars_id = $row['id'];

$sql = "INSERT INTO movies_stars (movies_id, stars_id) "

. "VALUES ('$movies_id','$stars_id')";

$mysqli->query($sql);

}

}

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

{

$genre = $mysqli->escape_string(trim($genres[$c]));

$sql = "INSERT IGNORE INTO genres (name) VALUES ('$genre')";

$mysqli->query($sql) or die($mysqli->error);

```

```

$genres_id = $mysqli->insert_id;

if ( $genres_id ){

$sql = "INSERT INTO movies_genres (movies_id, genres_id) "
. "VALUES ('$movies_id','$genres_id')";

$mysqli->query($sql);

}

else {

$sql = "SELECT id FROM genres WHERE name='$genre'";

$result = $mysqli->query($sql) or die($mysqli->error);

$row = $result->fetch_assoc();

$genres_id = $row['id'];

$sql = "INSERT INTO movies_genres (movies_id, genres_id) "
. "VALUES ('$movies_id','$genres_id')";

$mysqli->query($sql);

}

}

}

?>

```

Preparing the MySQL Database

In the screenshot below you see the basic syntax for inserted, selecting, updating, and deleting data from a MySQL table. Note than in every case the values are surrounded by single quotations.

INSERT

```
INSERT INTO table ( col1, col2 ) VALUES ( 'val1', 'val2' )
```

SELECT

```
SELECT * FROM table WHERE col1 = 'val1' AND col2 = 'val2'
```

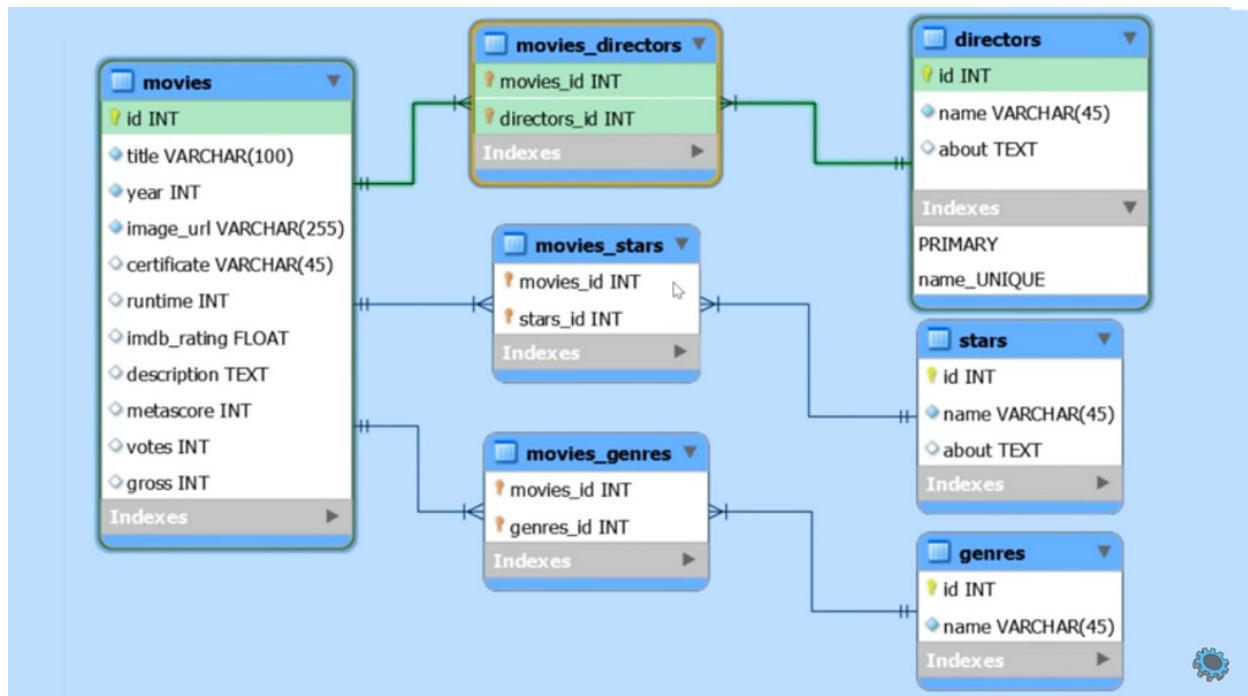
UPDATE

```
UPDATE table SET col1 = 'val1', col2 = 'val2' WHERE col3 = 'val1'
```

DELETE

```
DELETE FROM table WHERE col1 = 'val1'
```

Using that basic syntax, we're going to be inserting the data we scraped from the IMDb website into a MySQL database. Later we'll format and display it. The tricky part of inserting this data is going to be keeping track of everything while inserting data into these many-to-many relational tables. We're going to be doing this by inserting the movieID as you can see below. All three of these tables have the moves_ID as well as the directors_ID (or stars_ID or genre_ID depending on the table). To keep track of which movie corresponds to which director, etc., we will use a special function called `mysqli insert_id`. This will get the last key of the movie ID which has been inserted into the database to keep track of which movie is associated with which ID.



If you don't already have the database created you can grab the source code from the previous video and then go to phpMyAdmin. Under Databases, create the database called movies, then under the SQL tab paste in the source code from the previous video and click go. You can see that it creates all the tables we will need.

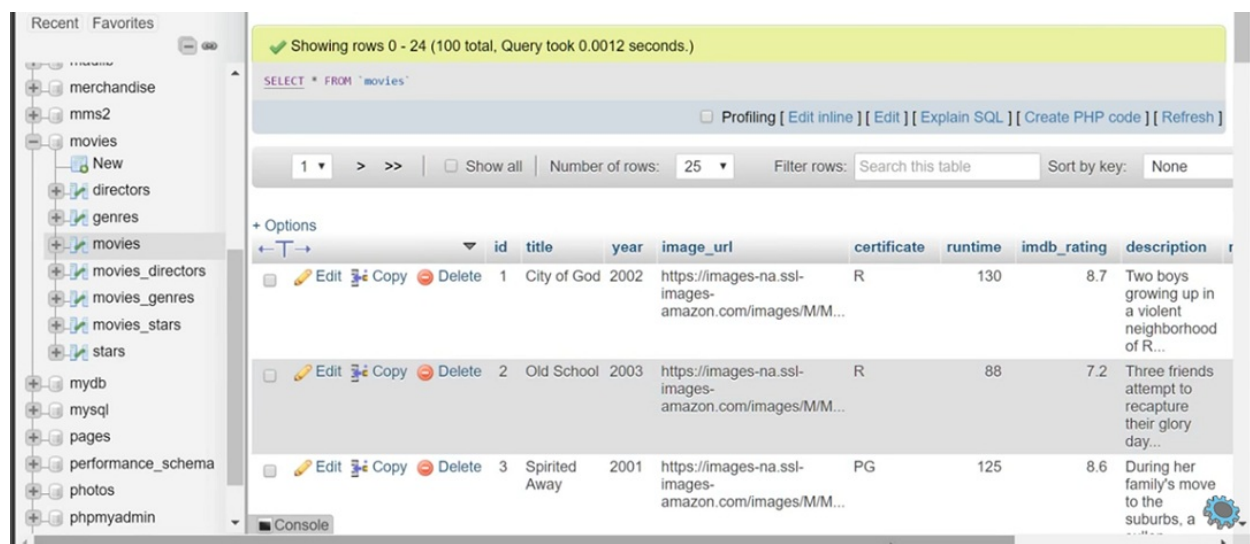
You also need to have the IMDb scrape function we created previously, so if you don't have that you can go to the video Web Scraping Using PHP and grab the source code from that. I recommend saving this in a new file and including it in the code we will write today.

Preview the End Result

Before I break down the code we're writing today, I want to show you guys what we're going to be accomplishing. If I run the script it scrapes all the movies from IMDb along with the associated movie data and inserts it into our database. If you click to view the page source, you can see below that all the data is nicely formatted in an array, making it really easy to insert the data into our MySQL database.

```
1 Array
2 (
3     [0] => Array
4     (
5         [title] => City of God
6         [year] => 2002
7         [image] => https://images-na.ssl-images-
amazon.com/images/M/MV5BMjA4ODQ3ODkzNV5BM158anBnXkFtZTYwOTc4NDI3._V1_UX67_CR0,0,67,98_AL_.jpg
8         [certificate] => R
9         [runtime] => 130
10        [genres] => Crime, Drama
11        [imdb_rating] => 8.7
12        [metascore] => 79
13        [description] => Two boys growing up in a violent neighborhood of Rio de Janeiro take different paths: one becomes a photographer,
the other a
14        [directors] => Fernando Meirelles, Kátia Lund
15        [stars] => Alexandre Rodrigues, Matheus Nachtergaele, Leandro Firmino
16        [votes] => 557059
17        [gross] => 7,563,397
18    )
19 )
```

After running the script, if we go back to our database and refresh the page we see that all of our tables are there. If you click on movies, for example, you see all the movie information is present.



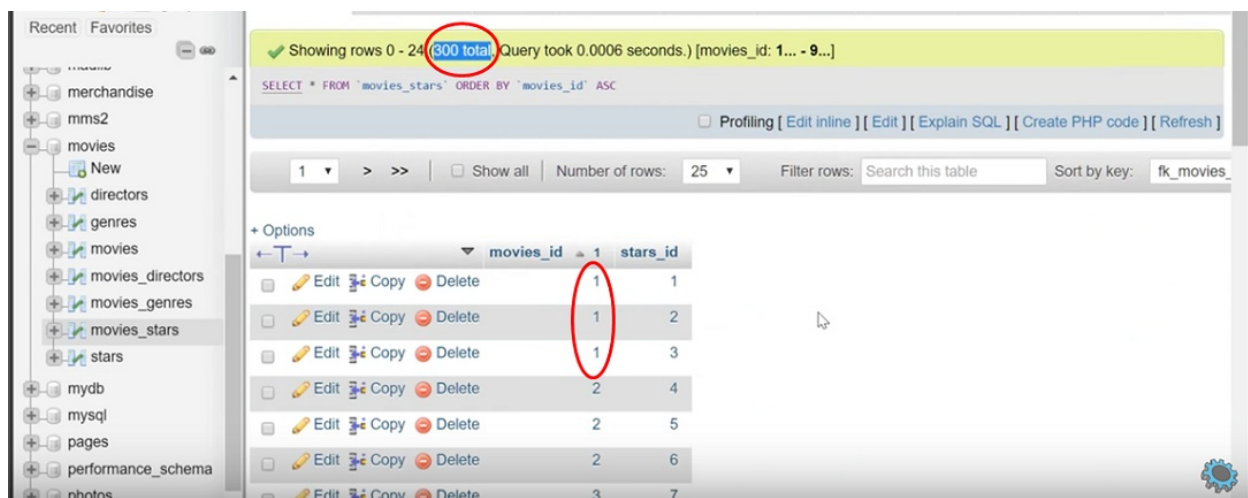
Showing rows 0 - 24 (100 total, Query took 0.0012 seconds.)

SELECT * FROM 'movies'

1 > >> Show all Number of rows: 25 Filter rows: Search this table Sort by key: None

	id	title	year	image_url	certificate	runtime	imdb_rating	description
<input type="checkbox"/>	1	City of God	2002	https://images-na.ssl-images-amazon.com/images/M/M...	R	130	8.7	Two boys growing up in a violent neighborhood of R...
<input type="checkbox"/>	2	Old School	2003	https://images-na.ssl-images-amazon.com/images/M/M...	R	88	7.2	Three friends attempt to recapture their glory day...
<input type="checkbox"/>	3	Spirited Away	2001	https://images-na.ssl-images-amazon.com/images/M/M...	PG	125	8.6	During her family's move to the suburbs, a

You can also check out the directors and stars and other information in this full database. One thing to note is that in the relational tables there are far more records. For the stars, for instance, there are 300. This is because one movie can have many different stars. In the movies_stars relational table shown below you can see that the first three stars all correspond to the first movie ID.



Connecting to the Database and Establishing Settings

The first thing to do when starting to code this is to include the source code for scraping IMDb. Use the include keywords to include that function and then store everything inside the variable called data. To call that function we need to include the parameters for years of movies (start and end) and pages for the search (once again start and end). I don't recommend scraping a lot of pages from the start as IMDb could ban your IP.

```
<?php
```

```
include('scrape_imdb.php');
```

```
$data = scrape_imdb(1999, 2017, 1, 2);
```

Now we're ready to connect to the database. Make sure you use your correct username and password, and then for the database enter movies because that is the database we'll be working with. Next, create the new mysqli connection using the class name and providing all the parameters we just set.

```
$host = 'localhost';
```

```
$user = 'root';
```

```
$pass = 'mypass123';
```

```
$db = 'movies';
```

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

Preparing Data for Insertion into Database

With the connection established, it's time to prepare the data to be inserted into our database. We'll use a for loop to loop through the array stored in the data variable. For the description, create a new description variable and use the escape_string function to remove any special characters that MySQL may not agree with. Do the same thing for the title.

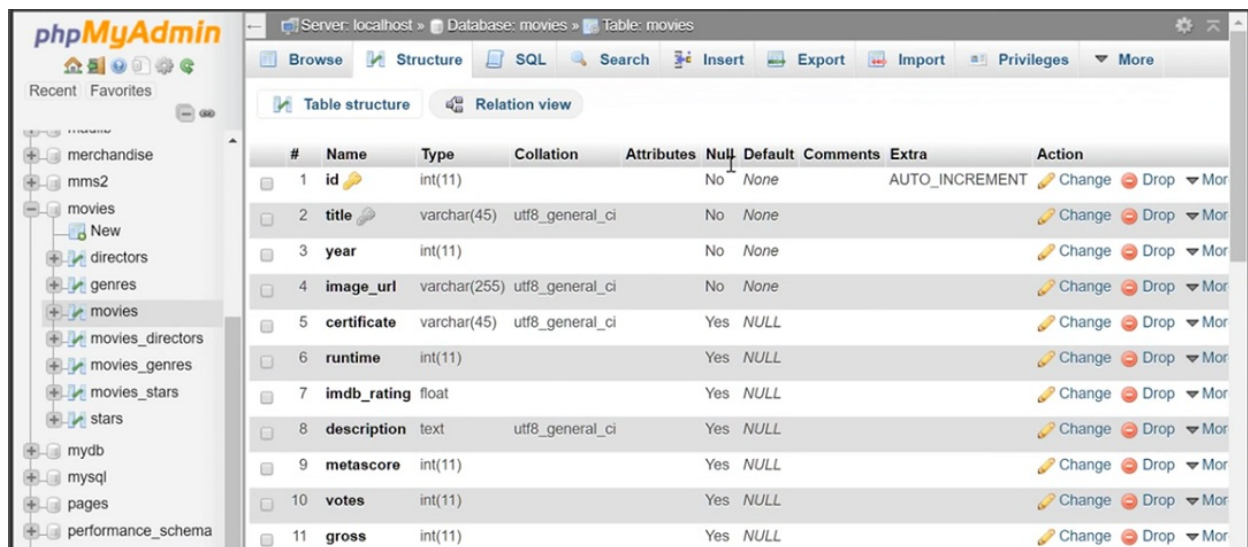
```
for ($i = 0; $i < count($data); $i++) {
```



```
$description = $mysqli->escape_string($data[$i]['description']);
```

```
$title = $mysqli->escape_string($data[$i]['title']);
```

For both votes and gross, we need them to be in integer format. You can check this by looking at the table in phpMyAdmin and seeing the column listing the data type as shown below. To make sure the values being inserted into the database are in the correct format, use the intval function. For the votes it is very straightforward, but the gross is a string including commas so it needs a little extra work to convert it to an integer. Use str_replace to replace the commas, then use intval as we did for the votes.



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)			No	None		AUTO_INCREMENT	Change Drop Mor
2	title	varchar(45)	utf8_general_ci		No	None			Change Drop Mor
3	year	int(11)			No	None			Change Drop Mor
4	image_url	varchar(255)	utf8_general_ci		No	None			Change Drop Mor
5	certificate	varchar(45)	utf8_general_ci		Yes	NULL			Change Drop Mor
6	runtime	int(11)			Yes	NULL			Change Drop Mor
7	imdb_rating	float			Yes	NULL			Change Drop Mor
8	description	text	utf8_general_ci		Yes	NULL			Change Drop Mor
9	metascore	int(11)			Yes	NULL			Change Drop Mor
10	votes	int(11)			Yes	NULL			Change Drop Mor
11	gross	int(11)			Yes	NULL			Change Drop Mor

```
$votes = intval($data[$i]['votes']);
```

```
$gross = intval(str_replace(',', '', $data[$i]['gross']));
```

Inserting Data into Movies Database

Using the basic syntax given in the beginning of this tutorial, we'll use the insert function to put all our data into the movies database. We'll store all of the SQL used to insert the data into our sql variable. The ignore keyword is used to make sure we don't insert duplicates. In the values you will notice there are curly brackets around the data. This is because we are using associative arrays and they must be there for the code to work.

```
$sql = "INSERT IGNORE INTO movies (title,year,image_url,certificate,runtime,
```

```
imdb_rating,metascore,description,votes,gross)
```

```
VALUES ('$title','{$data[$i]['year']}','{$data[$i]['image']}',
```

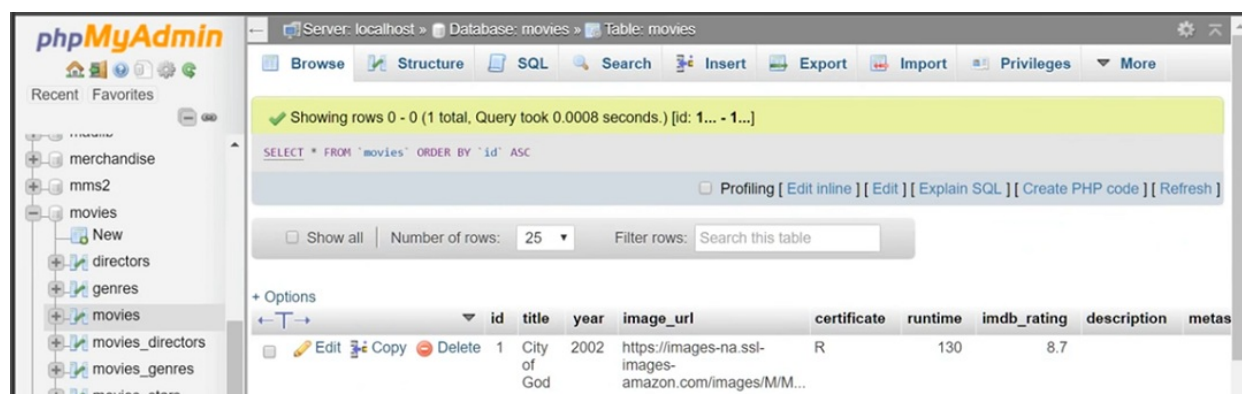
```
'{$data[$i]['certificate']}','{$data[$i]['runtime']}','{$data[$i]['imdb_rating']}',
```

```
'{$data[$i]['metascore']}','$description','$votes','$gross')";
```

Now that we have the query, we're ready to execute it by calling mysqli->query with our sql variable as a parameter. In case it has an error causing it to fail, we want to use the die function as well.

```
$mysqli->query($sql) or die($mysqli->error);
```


To test out the insertion, clear and recreate the database . To test with just one movie, terminate the script before the end of the loop. If you go back into the movies database you can see the information has been inserted.



I mentioned before that we would be using the `insert_id` function to keep track of which movie had been inserted last, so we make sure all our relational tables keep the right movie associated with each director and star. Set a `movies_id` variable using this function to get the key of the last movie record inserted. If no record is inserted, for example if it is a duplicate, this function will return zero. If this happens, we want to jump back to the beginning of the loop without doing anything more. To do this, use `continue`.

```
$movies_id = $mysqli->insert_id;
```

```
if ( !$movies_id ) {
```

```
    continue;
```

```
}
```

Inserting Data into Relational Tables

Looking back at our screenshot from earlier where we looked at the format of our data, we see that the directors, stars, and genres are stored in our array as comma-separated string values. We want to break those values down and store them in their own individual arrays.



We'll focus on formatting the directors since the stars and genres are done exactly the same. First we want to use the `explode` function with a comma as the delimiter to create different arrays out of the string values.

```
$directors = explode(",", $data[$i]['directors']);
```

```
$stars = explode(",", $data[$i]['stars']);
```

```
$genres = explode(",", $data[$i]['genres']);
```

Make another loop with a new counter to loop through the directors within each movie. We'll format the directors by storing them inside the director array using the `escape_string` function to remove any spaces that may be there after using the `explode` function. In the directors table, all we need is the ID and name. The ID is inserted automatically because it is auto incremented so really all we need is the director's name. We insert this and execute it similarly to our previous information. Once again we use the `insert_id` function to keep track of the last one inserted.

```
for ($c = 0; $c < count($directors); $c++)
```

```
{
```

```
$director = $mysqli->escape_string(trim($directors[$c]));
```

```
$sql = "INSERT IGNORE INTO directors (name) VALUES ($director)";
```

```
$mysqli->query($sql) or die($mysqli->error);
```

```
$directors_id = $mysqli->insert_id;
```

Now that we have the `movies_id` from the first `insert_id` function and the `directors_id` from the `insert_id` function we just wrote, we're ready to insert that information into the `movies_directors` relational table.

If the `directors_id` variable is not false, we insert the `movies_id` and `directors_id`. We don't use the `IGNORE` keyword here because there can be duplicates.

If `directors_id` comes back false, that means the director has already been inserted into the directors table. In this case, we need to get the director ID in an alternate method. To do this we'll find the director name in our director table by using the `SELECT` statement.

```
if ( $directors_id ){
```

```
$sql = "INSERT INTO movies_directors (movies_id, directors_id) "
```

```
. "VALUES ('$movies_id', '$directors_id')";
```

```
$mysqli->query($sql);
```

```
}
```

```
else {
```

```
$sql = "SELECT id FROM directors WHERE name='$director'";
```

```
$result = $mysqli->query($sql) or die($mysqli->error);
```

```
$row = $result->fetch_assoc();
```

```
$directors_id = $row['id'];
```

```
$sql = "INSERT INTO movies_directors (movies_id, directors_id) "  
.  
"VALUES ('$movies_id','$directors_id')";  
  
$mysqli->query($sql);  
  
}  
  
}
```

Now we're inserting the data inside the movies, directors, and movies_directors by tracking the last ID of movies and directors. The stars and genres are done exactly the same so I won't elaborate further on that. You can grab the full source code from above or just copy your directors code and replace directors with stars and genres.

I hope you enjoyed this tutorial. Be sure to come back next time when we'll be displaying all this data in a nicely formatted way. As always, if you found this tutorial helpful please like, share, and subscribe to Clever Techie!