# Lab Exercise:

This document is a walkthrough through the process of **creating and using databases**. After following all steps you will have database with two tables, which contain some data.

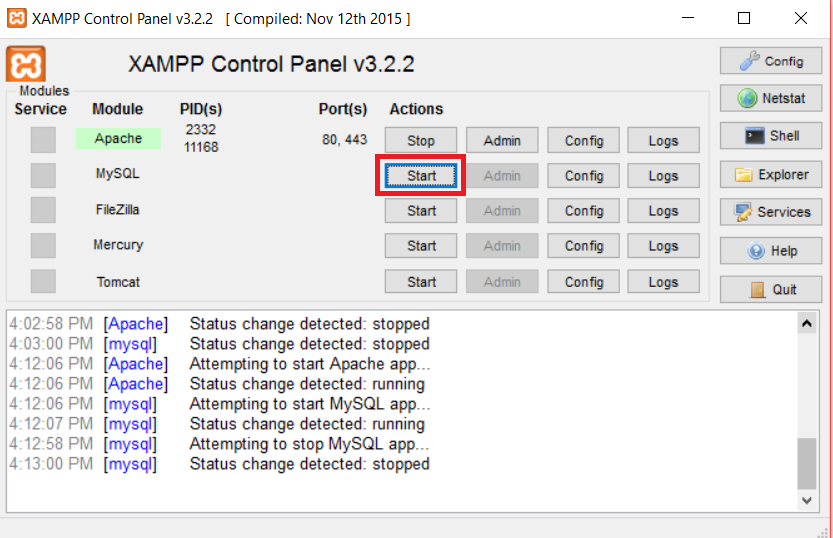
This lab is part of the [“Software Technologies” course @ SoftUni](https://softuni.bg/courses/software-technologies).

# Part I: Create MySQL Database

This first part will show you how to **create your first MySQL DB**.

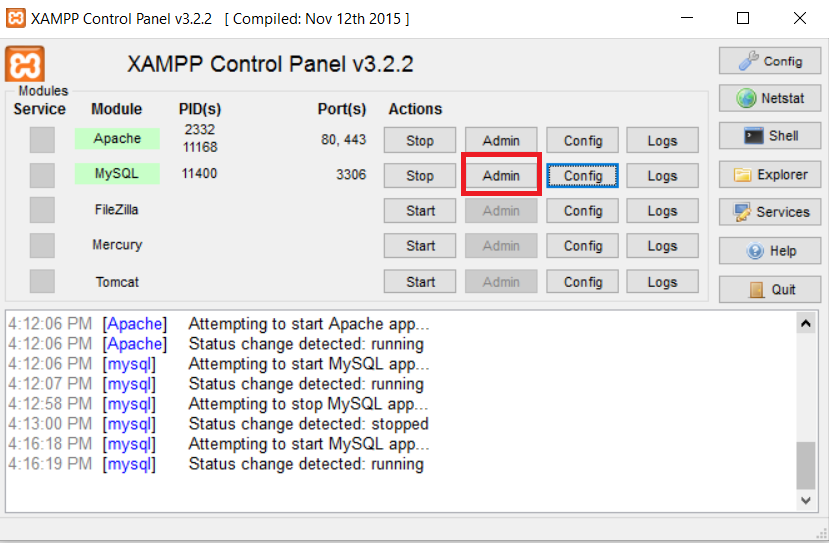
## Start MySQL Database Server

Now we need to start the **MySQL module in XAMPP**:



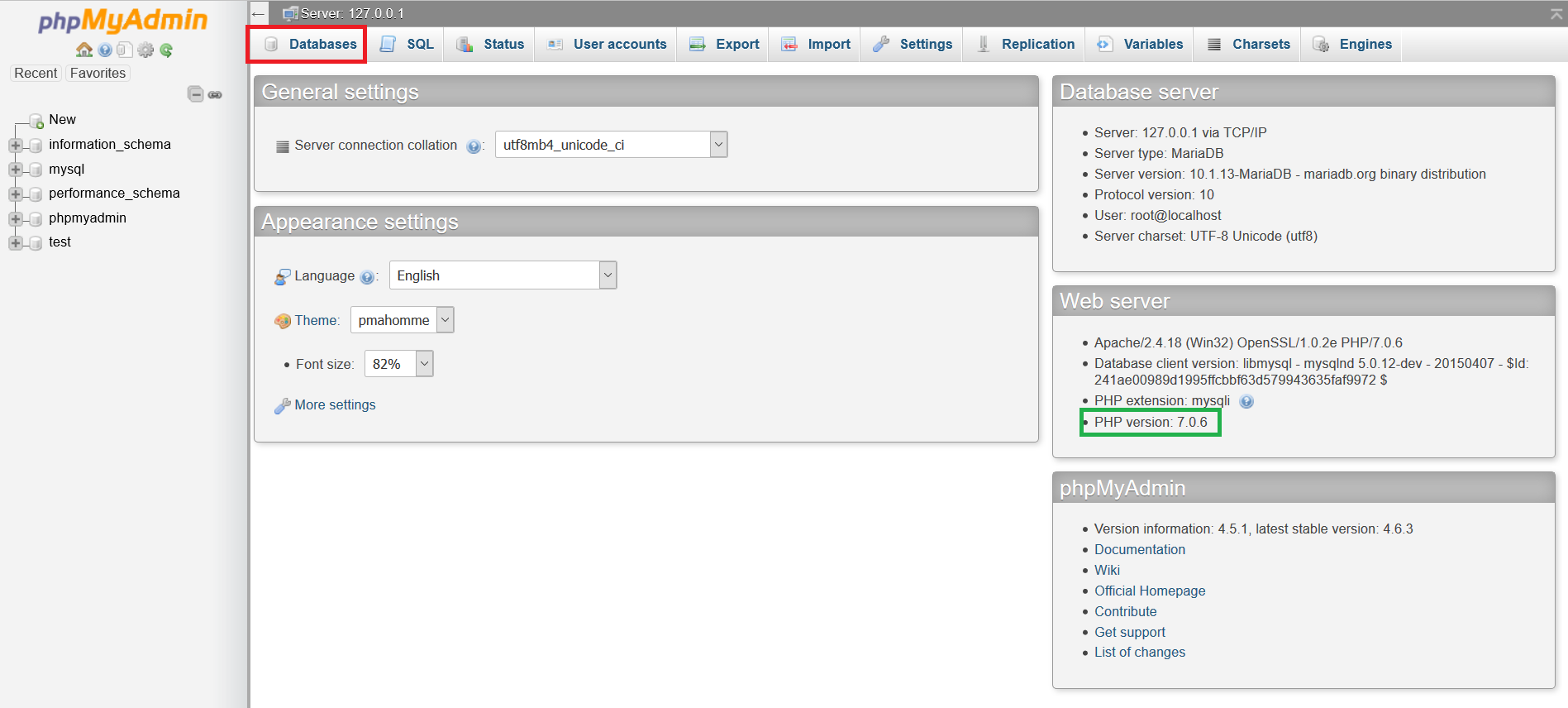
## Go to phpMyAdmin Panel

In order to create a MySQL database, we need to go to the **phpMyAdmin** page:

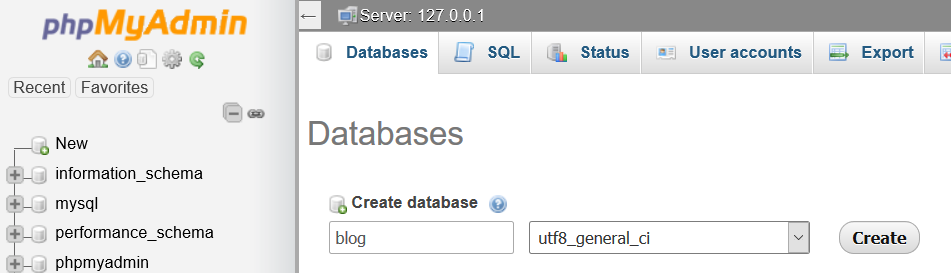


## Create a Database

In order to create a DB, we need to go to the **phpMyAdmin** **databases** page:



On the next window fill the required fields and click the **[Create]** button. Change the encoding to **utf8\_general\_ci**. This encoding will support Cyrillic and other non-Latin languages. You should have something like this:

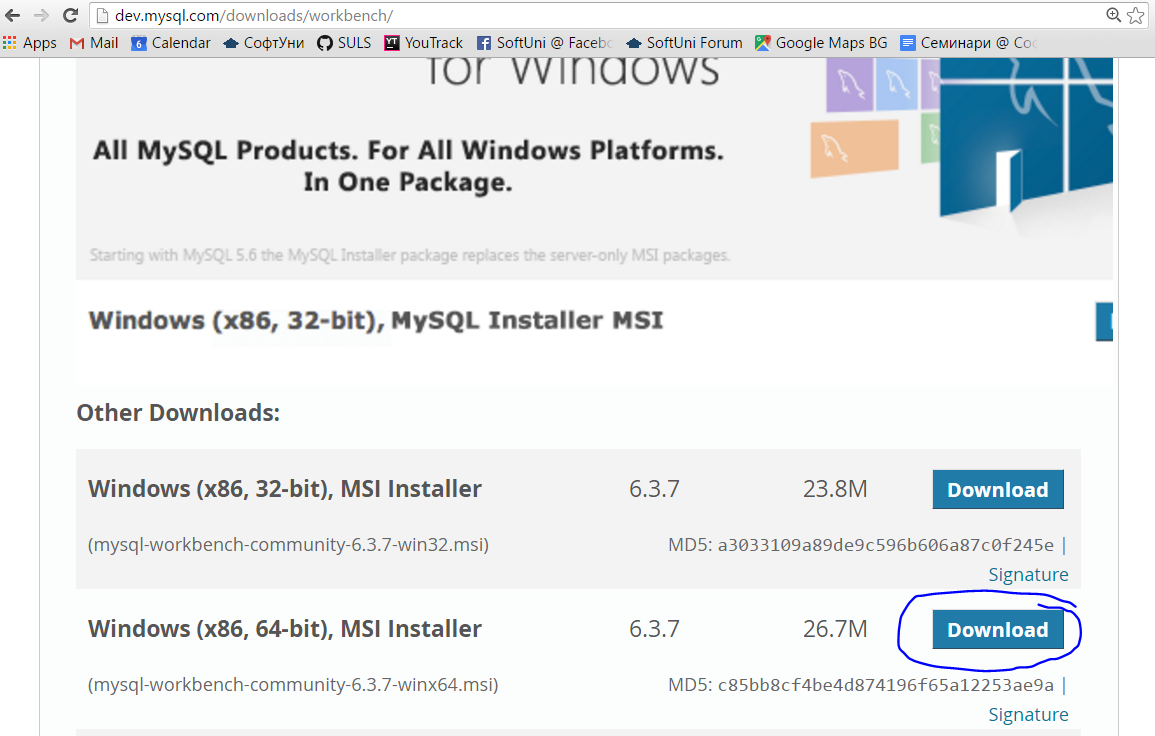


Make sure to remember the **DB name**, because we will need it later on.

# Part II: Manipulate MySQL Database with MySQL Workbench

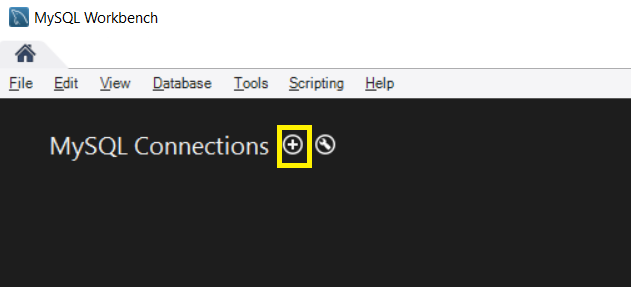
This part will show you how to **manipulate a database with MySQL Workbench.**

If you do not have MySQL Workbench, go to <http://dev.mysql.com/downloads/workbench/>, download the MSI Installer, and install MySQL Workbench.

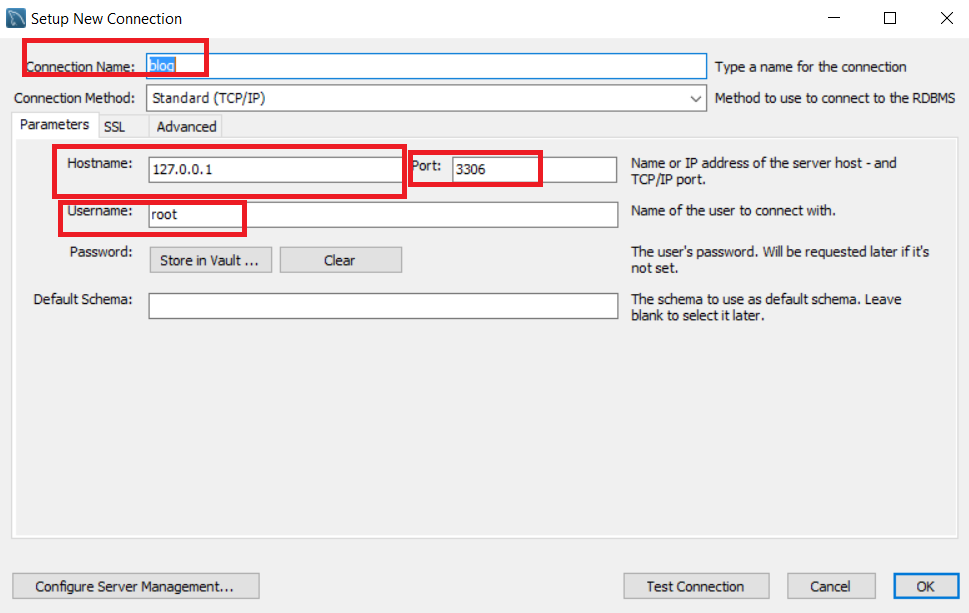


## Connect the Database

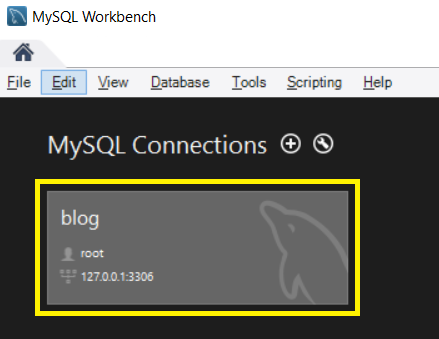
On the home screen click on the white plus:



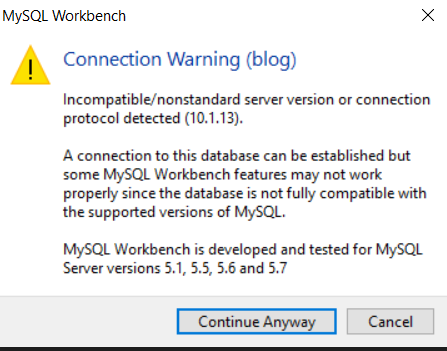
A new window will open, that will require the connection information. Make sure you enter the correct port that XAMPP uses for MySQL:



Click OK and you should be returned to the home page.



Open the connection. You **might** **see** **warning** like this one, but just click on ‘Continue Anyway’:

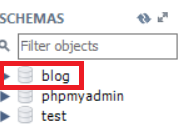


Now you will see a big screen, with open sidebars and menus. You should click on those buttons:

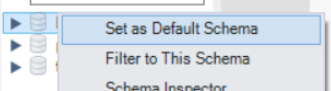


They are located in the top right corner and they will hide the right sidebar, and the bottom bar.

With that done we should locate our database in the left sidebar:

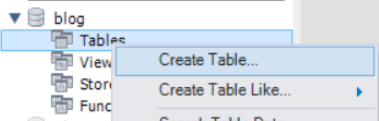


Right click on the DB and choose the option “Set as default schema”:

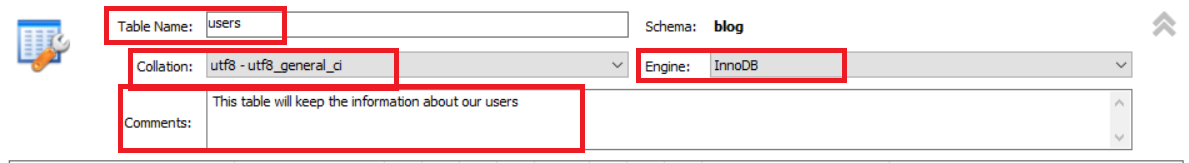


## Create Users Table

Expand the database. You should see a tables tab. Right click on it and select “Create table”.

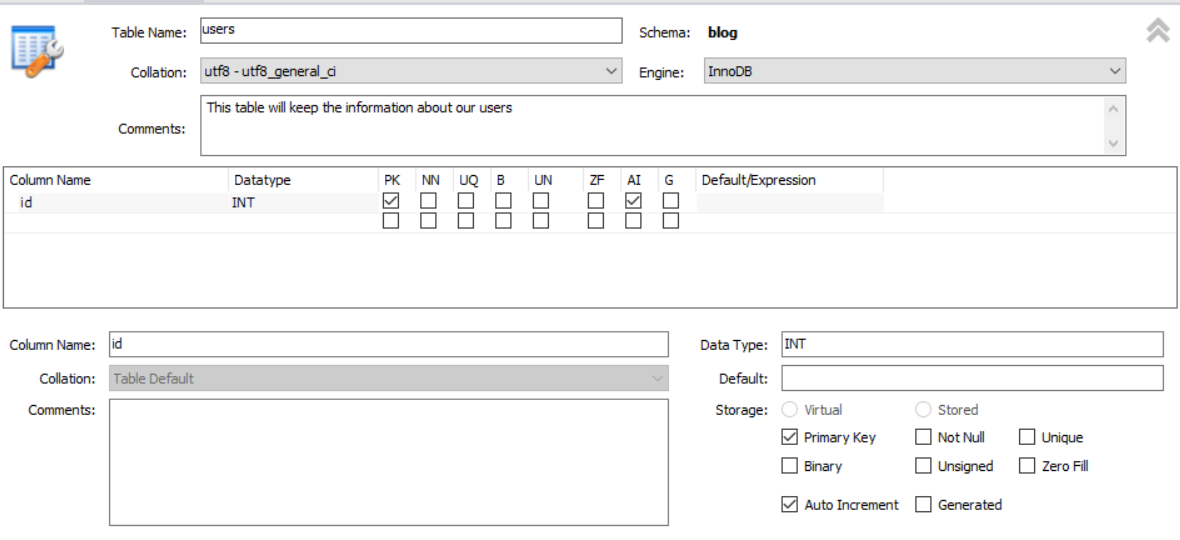


In the new window you need to give your table a name and description:



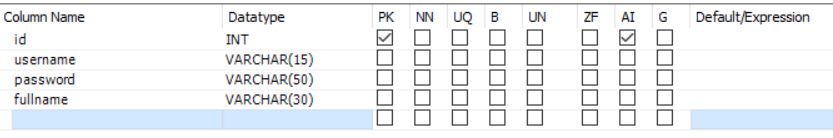
**Don’t** change the engine, but **change** **the** **collation**, to the **same** **collation** that we’ve **used** in our **DB**.

Let’s create some columns. Double click under the column tab. Set the column name to ‘id’ and the datatype to INT. Place checkmarks on PK (primary key) and AI (auto-increment). You should have something like this:

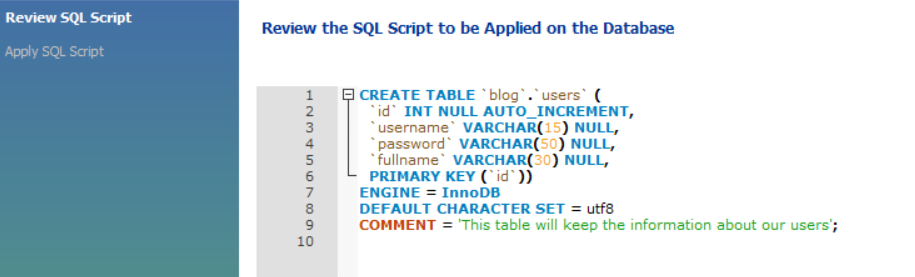


Let’s create the rest of the columns:

* **username** – **VARCHAR (15)**
* **password** – **VARCHAR (50)**
* **fullname** – **VARCHAR (30)**



Click on [Apply]. You should see this SQL script:



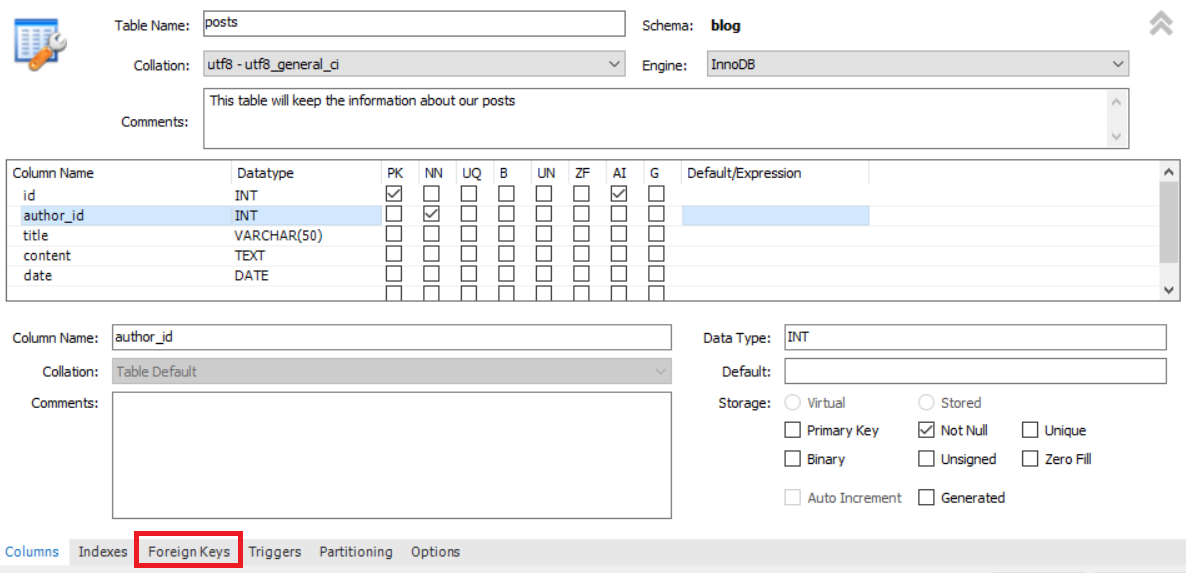
Click [Apply] and then [Finish].

## Create Posts Table

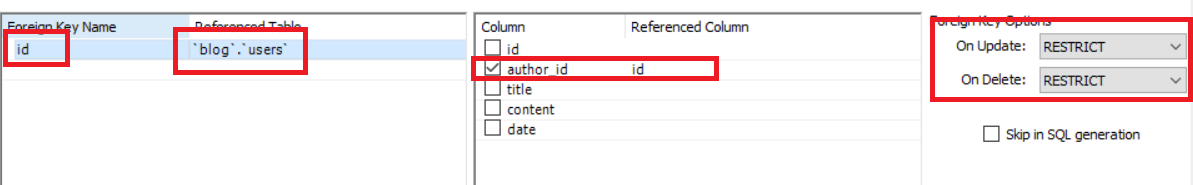
Let’s create another table. Our table will be called ‘posts’ and will keep the information for our posts. The **primary key** will be ‘id’, which should be ‘int’ with **auto increment**. The rest of the columns are:

* author\_id – int – Not Null
* title – VARCHAR (50)
* content – TEXT
* date – Date

It should look like that for now:

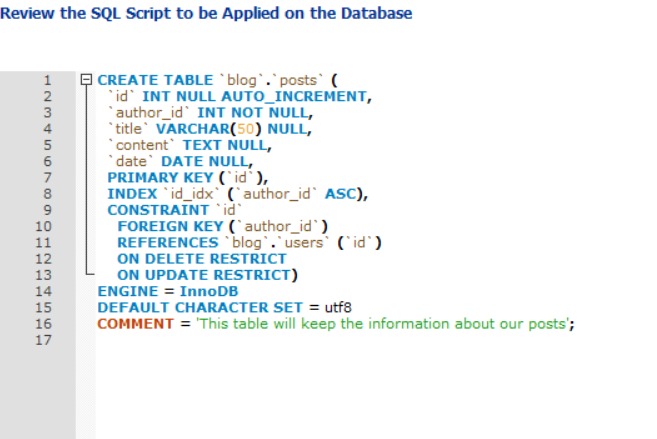


Let’s go to the ‘Foreign Keys’ tab:



Here we’ve created relation between our 2 tables. Now every time we create a post, there will be user linked to it.

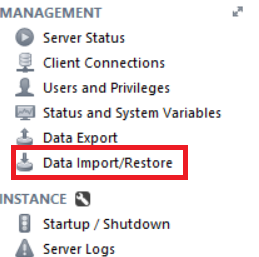
We should have this SQL script:



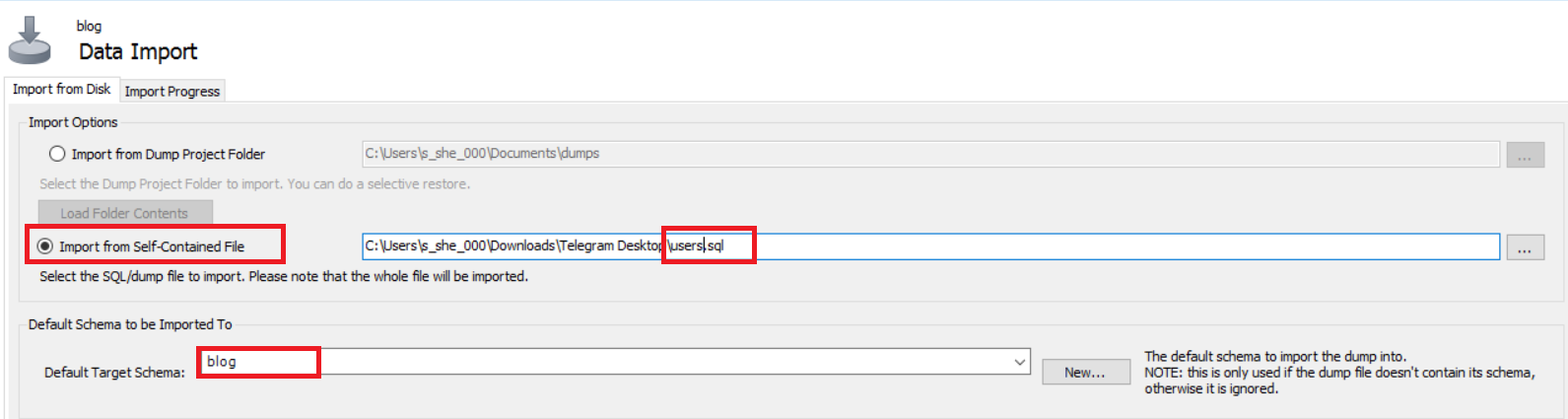
Click ‘Apply’ and then ‘Finish’.

## Import the Tables Data

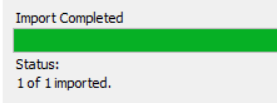
Go to the “Data Import/Restore” tab on the right:



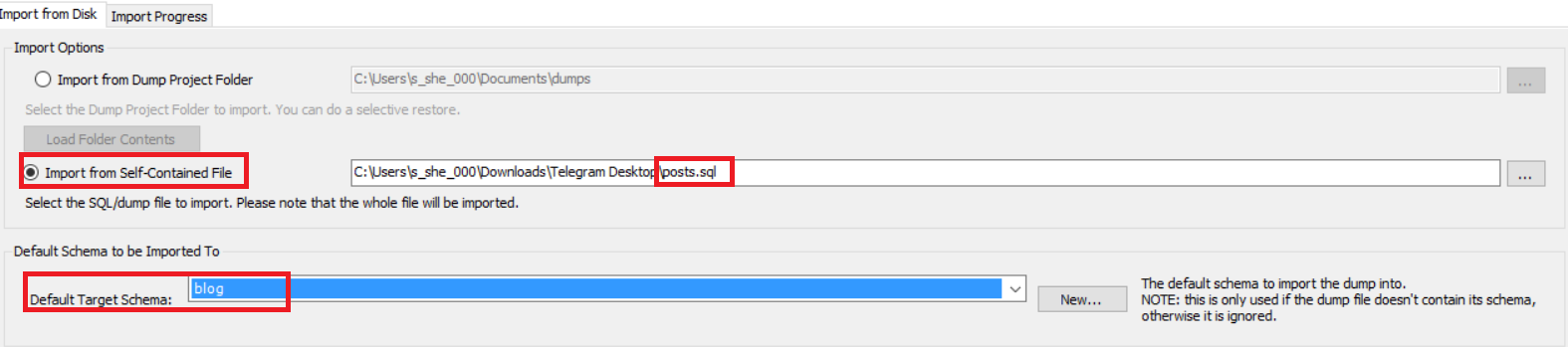
In the menu, choose ‘Import from Self-Contained File’. Change the ‘Default Target Schema’ to our DB. Import the ‘users.sql’ file, that you’ve received with this file, and click on ‘Import’



You should see this progress bar:



Let’s do the same thing, this time for the posts:



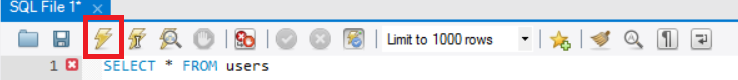
If everything worked correctly, you will have 5 users and 7 blog posts in your tables.

# Part III: Queries with MySQL Workbench

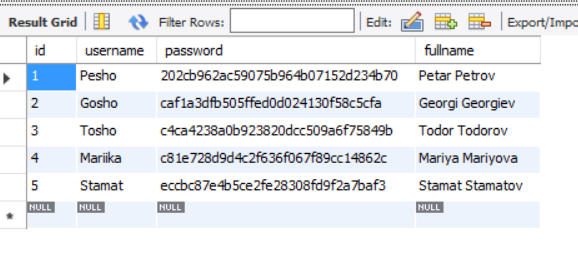
## SELECT Commands

Now let’s play with our tables. Go to **File** -> **New Query Tab**. This will open new window in which you can create queries. Let’s start with something simple. We will start with **SELECT** queries. They retrieve data from our tables, and give us the information we need. Let’s get all users from our users table:

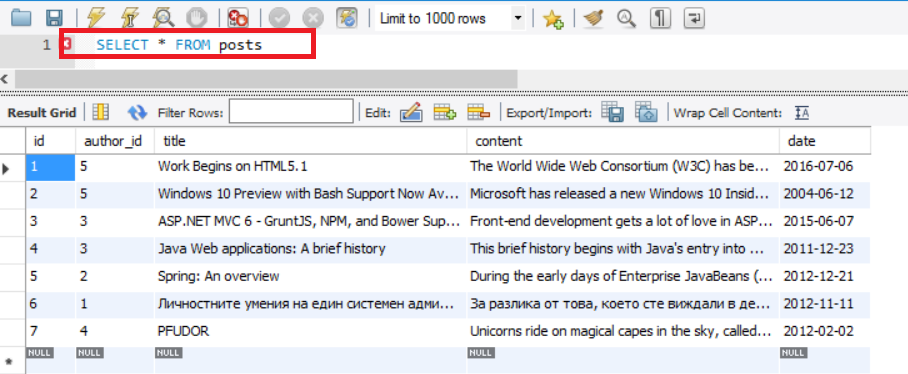
Enter ‘SELECT \* FROM users’ and click on the yellow lightning.



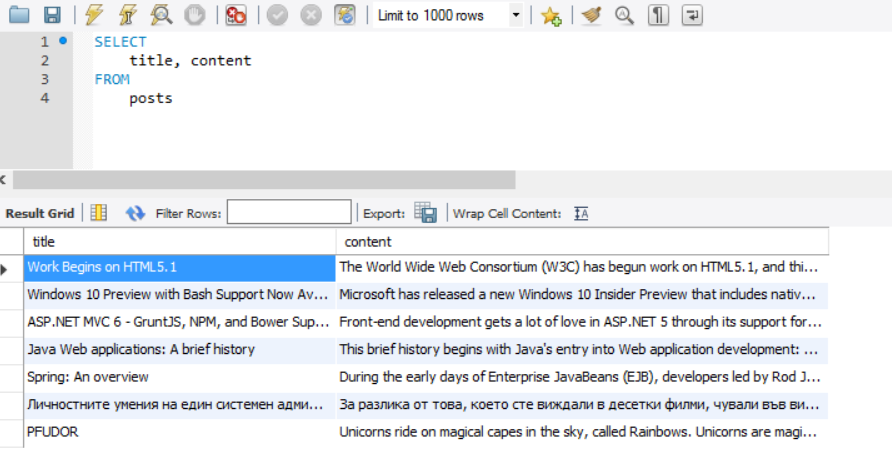
You should see this as a result:



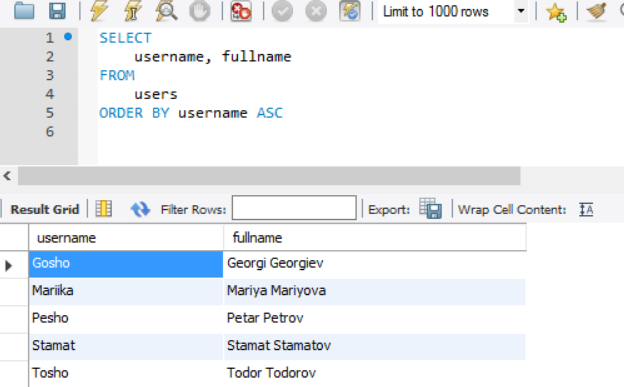
Let’s get all posts now:



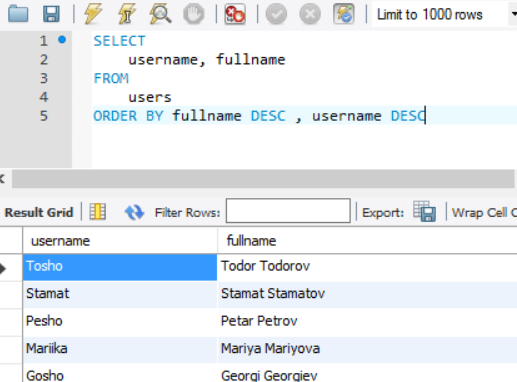
Let’s do something harder. We will get only the title and content from every post, ignoring the other fields:



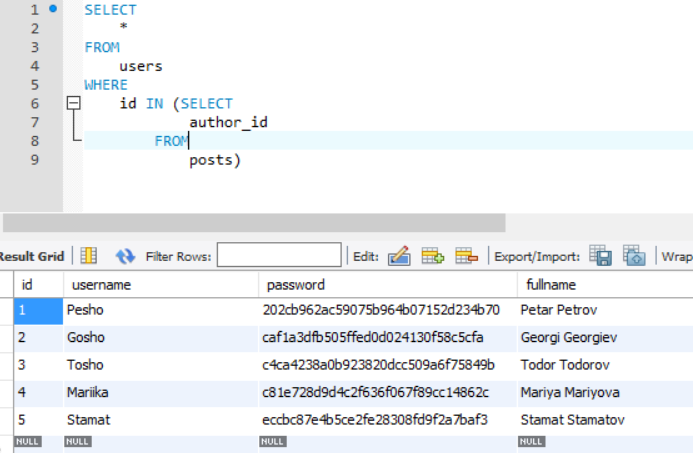
Now we will order the data in ascending order. Let’s take all usernames and full names of our users.



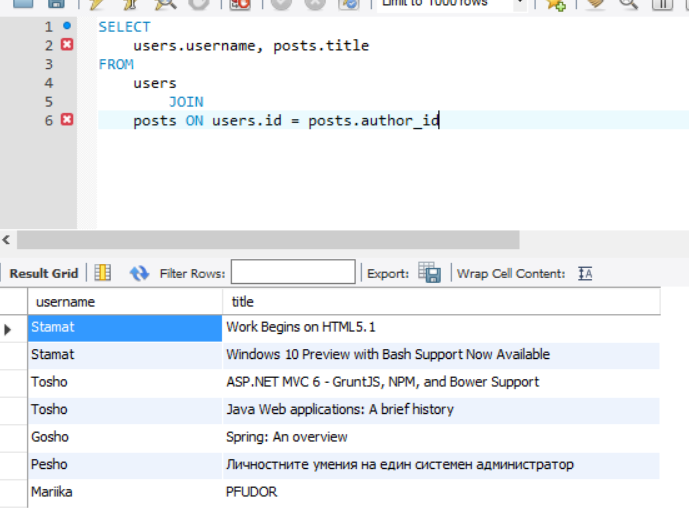
What if we have same username for 2 users? We should order by 2 parameters, but this time we will do it in descending order:



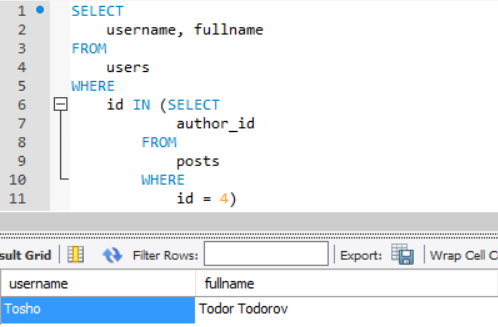
Let’s imagine that we’ve had many users, but not all of them have created blog posts. How can we get only the users that have created a blog post? Just like that:



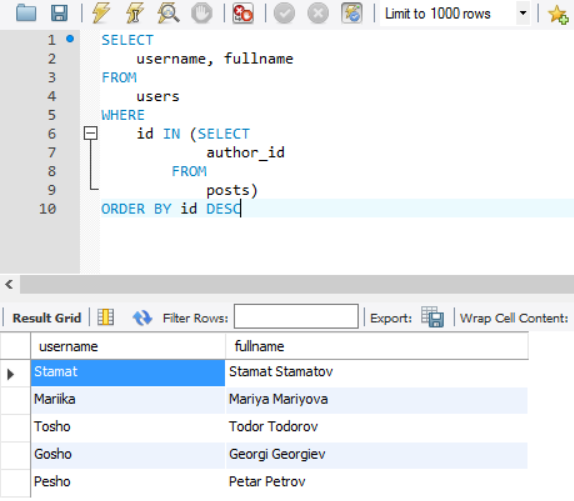
Something more interesting now. Let’s get the usernames with the post names for each user. The query should look like this:



Let’s get the creator of post with id=4 now:



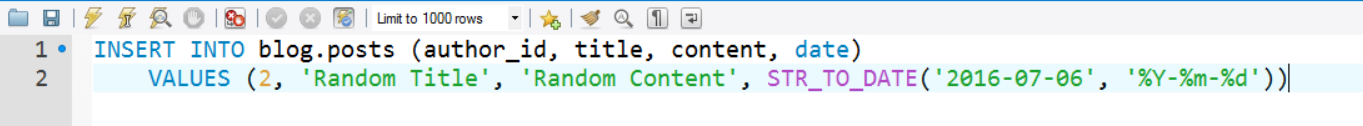
Let’s do one last **SELECT** query. It should get all of the post authors and order them by id in descending order. We will need only the username and full name.



How did we do that? To explain that I will split the query in 3 parts. The first part will be the nested **SELECT**. This query will return collection (array) with every ‘user id’ that has at least one post in the **posts** table. Let’s examine the first **SELECT** now. It takes every username and full name of the users, which ‘id’ is contained in the collection that our nested query returned. In other words, if we have post creators with ‘id’ – 3, 8 and 53 the query will do contains for every user id that we have (example -> collection.contains(currentId)). After we do that, the final part is the sorting by ‘id’ in descending order.

## INSERT Command

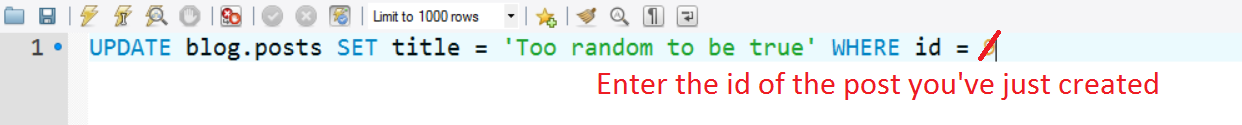
**INSERT** queries are used to enter data to our table. Let’s try them!



This query should create new post in our table. Create **SELECT** query to see if it worked.

## UPDATE Command

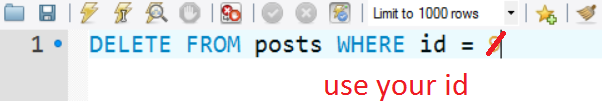
**UPDATE** queries are used to change the data in our tables.



This query should update our newly created post. Create **SELECT** query to see if it worked.

## DELETE Command

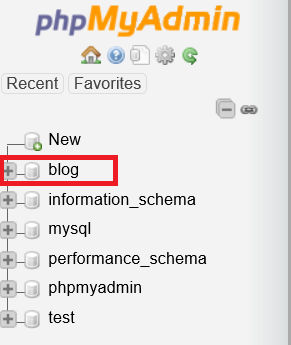
**DELETE** queries are used to delete data from our tables.



This query should delete our newly created post. Create **SELECT** query to see if it worked.

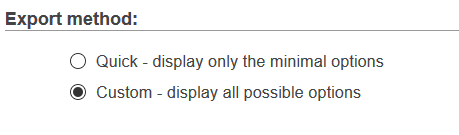
## Create Backup

We should create backups of our tables, because we are going to try a dangerous query. First of all, go to the phpMyAdmin page and click on our DB.

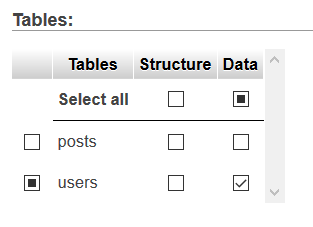


Now go to the ‘Export’ tab:

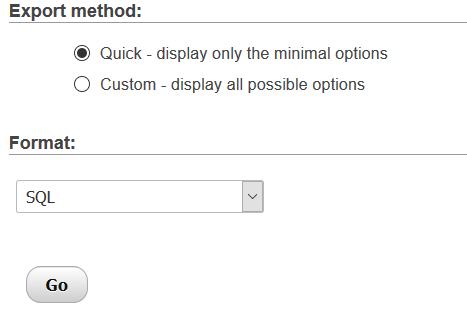




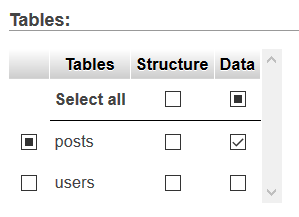
First thing we are going to do is change the export method to ‘Custom’. That will open more options that we can play with. However, we are only interested in removing the checkmark from the ‘Structure’ column in the ‘Tables’ menu:



In addition to that, remove the ‘Data’ checkmark from posts. That way we are only going to export the users from our table. Click on the ‘Quick’ export method, and then click on ‘Go’. Save the file somewhere.



Let’s go back to the ‘Custom’ export method again and switch the ‘Data’ checkmark on ‘posts’ and remove the one from ‘users’:



Go back to the ‘Quick’ method and export this table as well. Save the file. If you’ve followed those steps, you’ve just created almost an exact copy of the 2 files, you’ve received with this file.

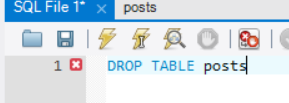
## DROP Command

I’ve said that we are going to do something dangerous. Here it is! We will delete our tables from the database, and not just **DELETE** them, but **DROP** them. What is the difference?

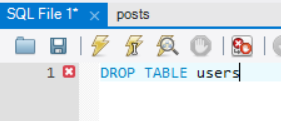
**DELETE** will delete the data but the table structure will remain the same and we can still rollback the data. Also with **DELETE** you can use the where condition i.e. to delete only certain records.

**DROP** will delete all data and the table structure as well.

Go back to the MySQL Workbench and write the following:



This will **DROP** our posts table. Let’s do the same with our users table:



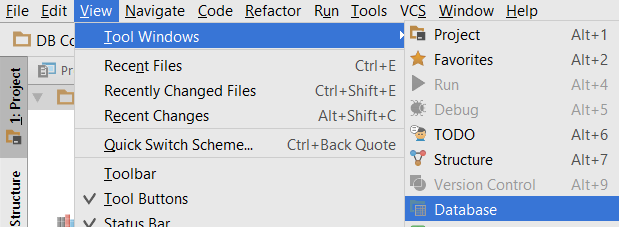
We have just deleted our 2 tables. We have no way to restore them… Or do we? We’ve created backup, remember? Don’t use it just now. We will need it later. Always be cautious when using **DROP**, **DELETE** and **TRUNCATE** queries. They are **dangerous** for your database.

# Part IV: Manipulate MySQL Database with PhpStorm

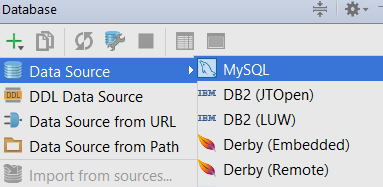
This part will show you how to **configure your DB with PhpStorm** and then manipulate it. You will see how to create tables and work with them.

## Connect the Database

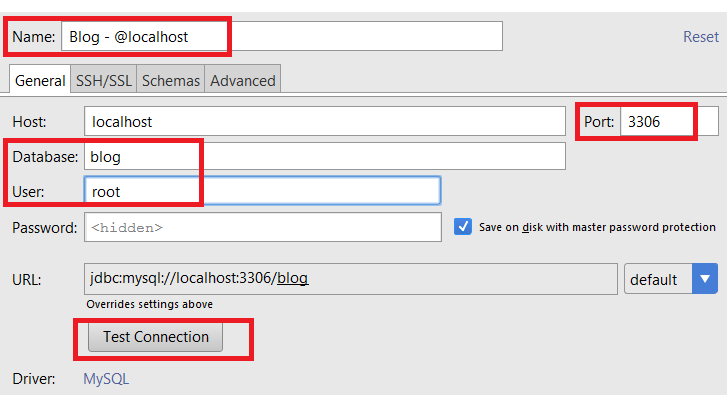
Create a new project. When you load up the home screen, you need to open the **Database** tool window. You can do that by going to **View** -> **Tool** **Windows** -> **Database**. This tool window allows you to connect to databases, create tables in them, create queries, view the relation diagrams and many more.



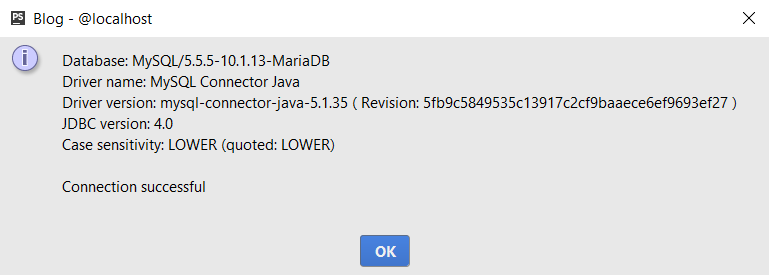
After the window opens, you need to click on the green plus, then choose **Data Source** -> **MySQL**:



You will be redirected to the MySQL Data source settings. You need to set up your database. Enter the name of the database you’ve created earlier, the port that XAMPP is using for the MySQL service and the default phpMyAdmin user, which should be ‘root’, if you haven’t changed anything.

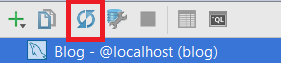


As a final step, you need to click on the “**Test Connection**” button. If everything is correct, you should see this page:



## Create Users Table

First you need to synchronize with your DB. Click on this icon in the DB Tool Window:



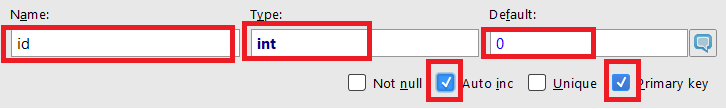
Now we need to **right click** on the DB, and go to **New** -> **Table**:



A window should popup. In this window you need to give your table a name and description. After you do that, click the green plus.



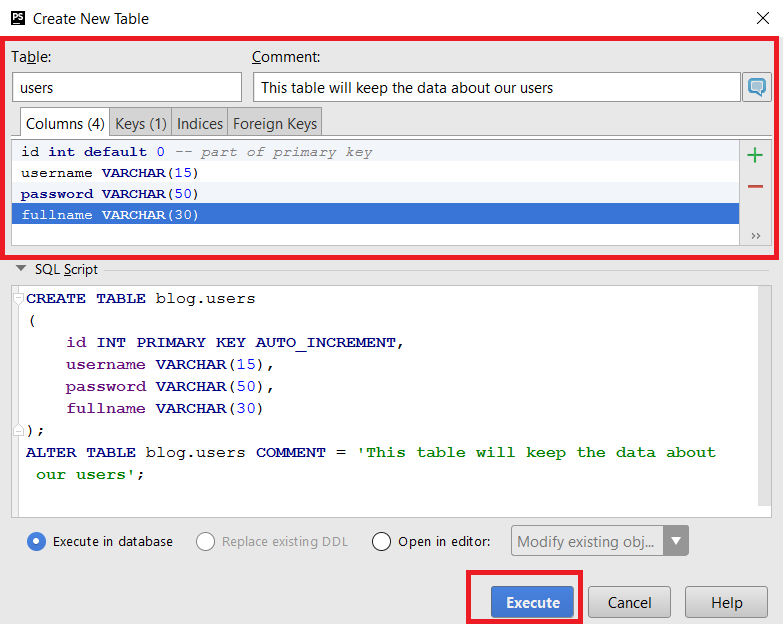
This should create your first column. **Every table** should have **primary key**. The primary key is column, which contains **unique values only**. In order to do that, we will create id column. That column will be **int**, which allows us to make it **auto increment**. That way, we’ve create a **unique user id column**, which will serve as our **primary key**.



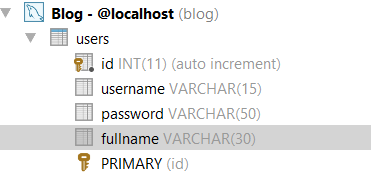
Now we need to create the rest of the columns, using the format **{name – type}**:

* username – VARCHAR (15)
* password – VARCHAR (50)
* fullname – VARCHAR (30)

In the end the window should look like this:



Right now in the DB Tool window you should see this:



## Create Posts Table

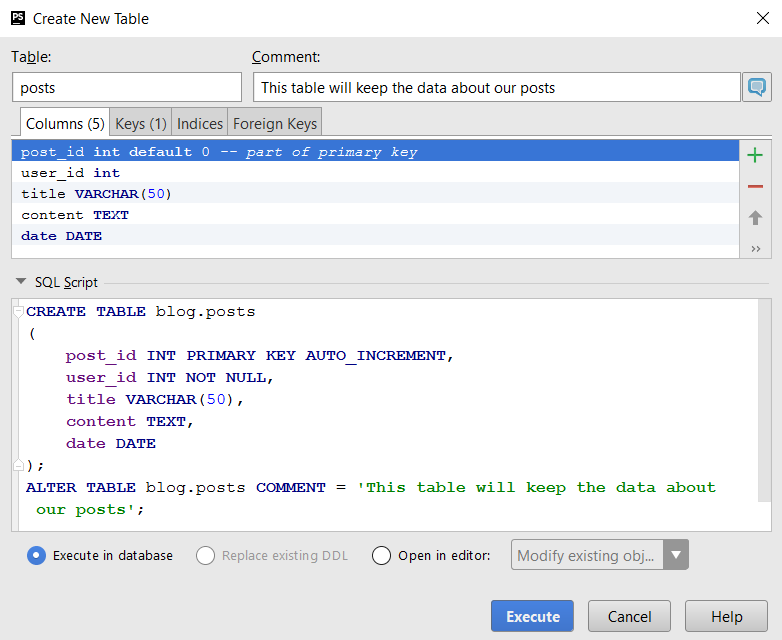
Let’s create another table:



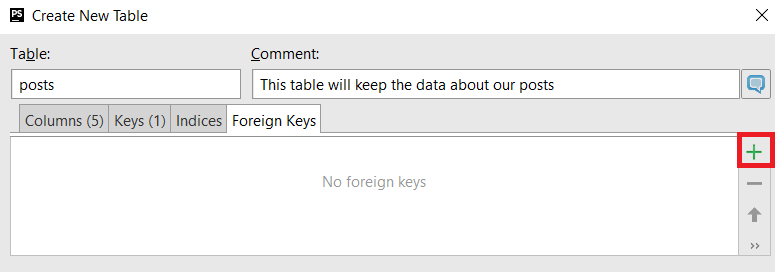
Our table will be called ‘posts’ and will keep the information for our posts. The **primary key** will be ‘post\_id’, which should be ‘int’ with **auto increment** and **default** value 0. The rest of the columns are:

* user\_id – int – Not Null
* title – VARCHAR (50)
* content – TEXT
* date – Date

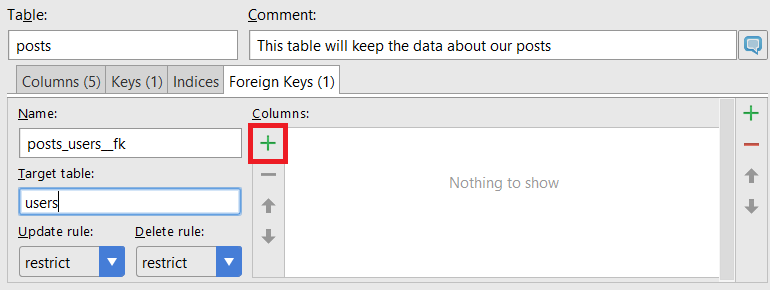
It should look like that for now:



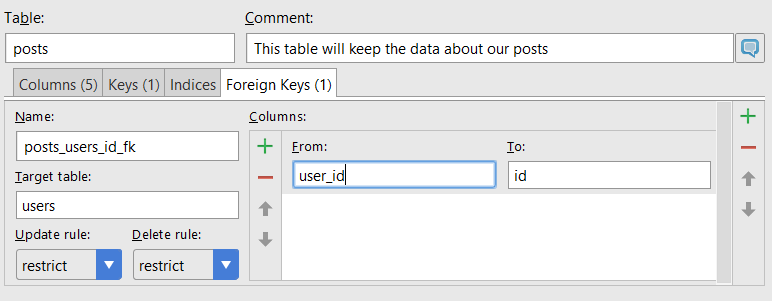
Let’s go to the ‘Foreign Keys’ tab:



Click on the green plus. In the new ‘**Target table**’ field, type ‘**users’** (the name of the other table).

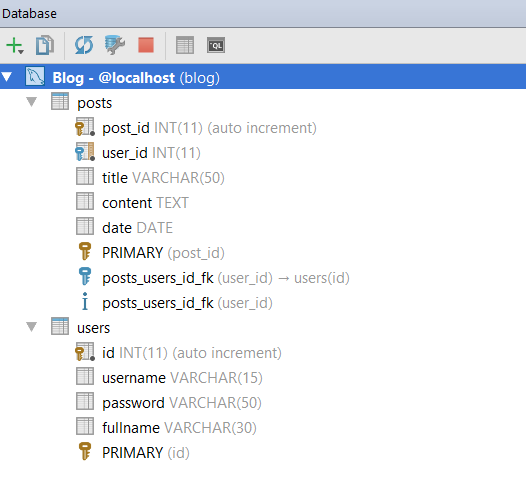


Add new column, using the green plus. In the ‘**To**’ field, enter the **primary key** of the **users** **table**, which is ‘id’. In the ‘**From**’ field, enter ‘**user\_id**’, which is our column in this table, that we’ve created earlier.



That is all for now, click ‘**Execute’**.

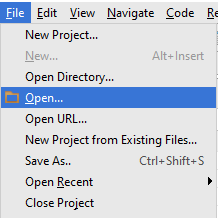
If you’ve done everything correctly, you should see this in the DB tool window:

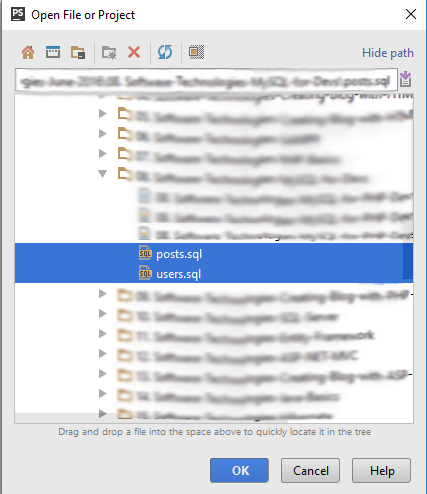


## Import the Tables Data

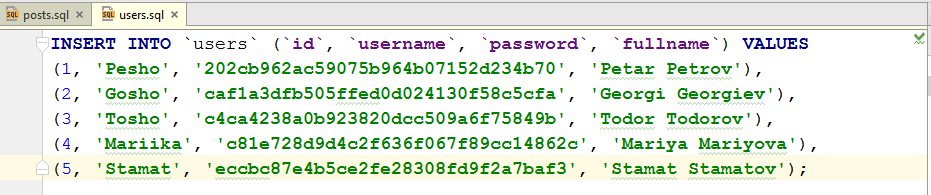
Same as before, we need to import data into the tables we’ve just created. Here, however we do not have a way to import them directly, so we will have to do it manually.

Open the two files **posts.sql** and **users.sql** with PHPStorm.

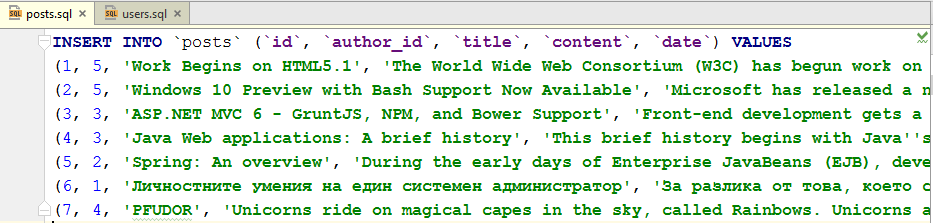




If you’ve opened the right files you should see the following:  
This



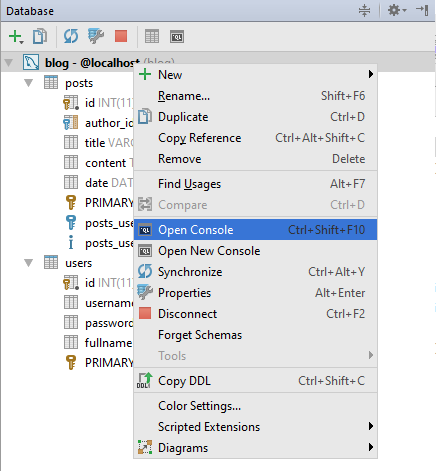
And this



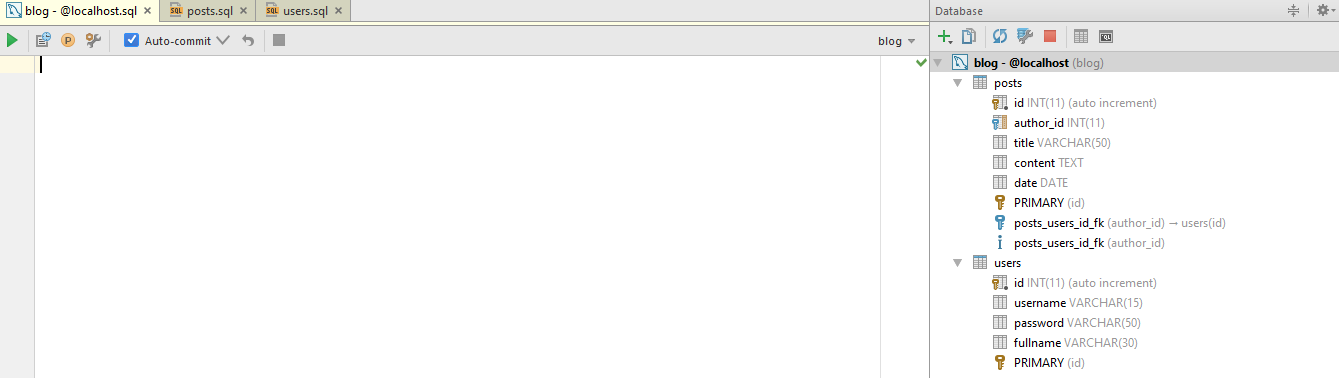
These are the two queries that will fill up our two tables.

Now, first we will have to fill the users table, because the posts table depends on it, due to the relation with the foreign key.

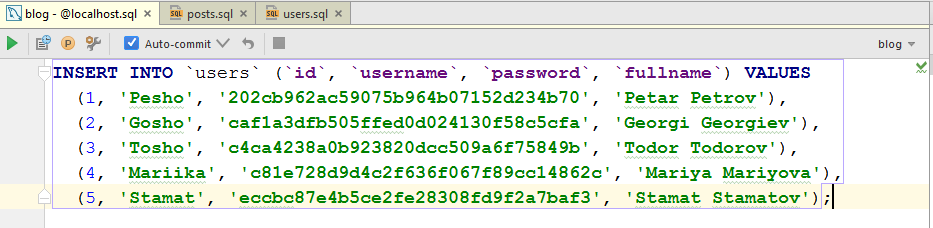
Open the database console, by right-clicking on the database in the explorer on the right, and selecting it.



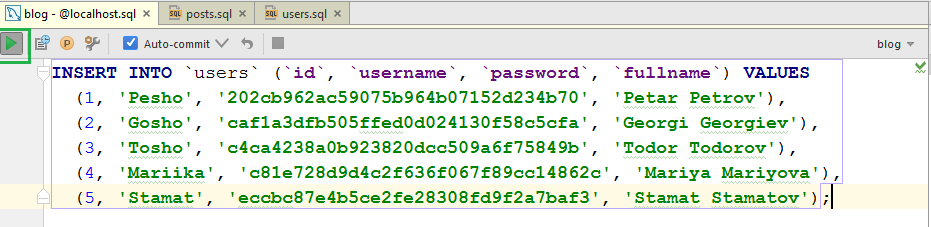
You should see the following:



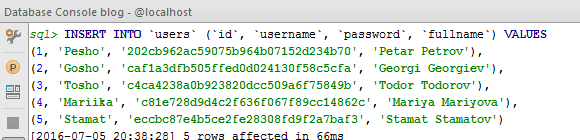
Now, copy the query from the **users.sql** file and paste it in the console.



And execute it.



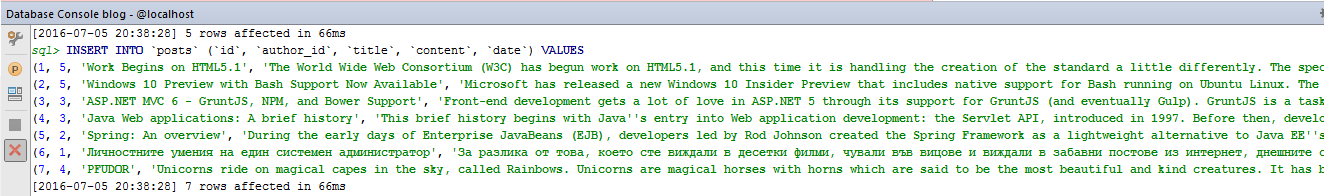
A console interface should popup from below showing you a success message, and affected rows in the table of the database.



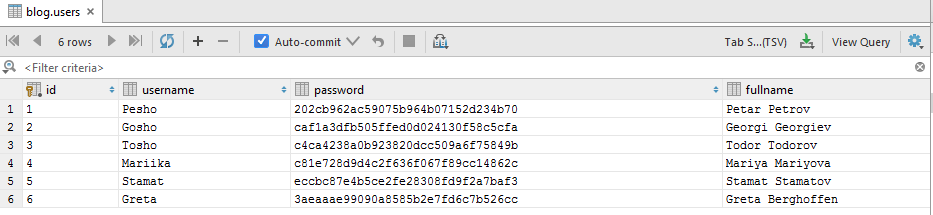
This means that the 5 users in the query have been registered successfully.

Now do the same with the query from the **posts.sql** file. Copy and paste the query, and execute it.

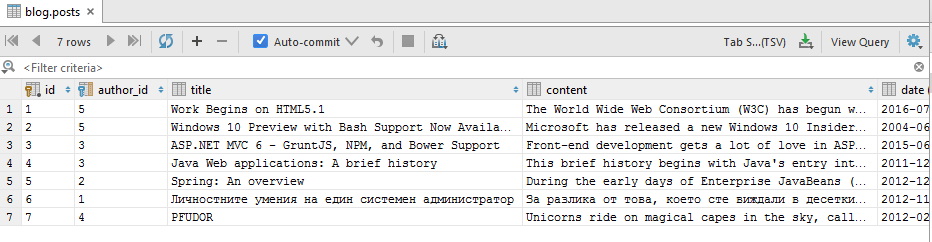
You should see the following:



There are 7 rows in the SQL query so 7 should be affected. With this we have filled our tables, and they should look like this now.



And the posts:

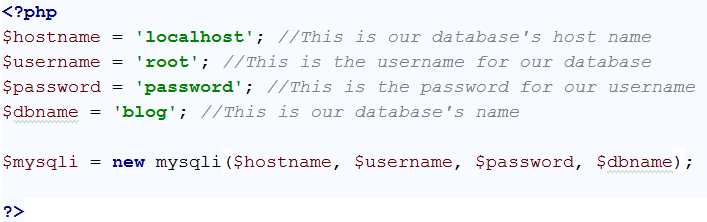


# Part V: Accessing MySQL with PHP, using mysqli

Now that we have our data set in our database, we can start manipulating it. We’ve seen how to do that with pure SQL, now let’s do it in PHP context, from scripts.

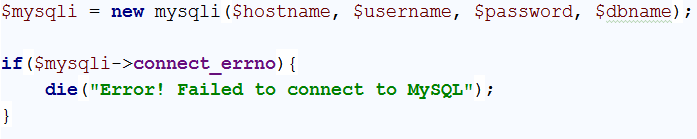
## Connecting to MySQL from PHP script

Use mysqli class to connect to the MySQL server and database.



Connecting to the database from a PHP script, using mysqli, requires a host name, a username, a password, and a database name. We use the data we’ve set for all 4 of those things, and then we make the connection.

## Checking the Connection Status



After connecting to the database, we must check if our connection was successful or has resulted in an error. Perform the simple check shown above.

## Setting the Default mysqli Charset

Before starting to work with queries and manipulating data, we must set the default charset to **UTF-8**, in case we have some special characters in the data stored on the database. This is done easily.

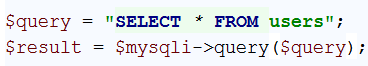


## Executing SQL Queries trough Existing MySQL Connection

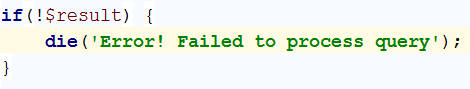
Finally – getting to the real business – executing SQL queries. The query syntax in PHP script is exactly the same. Executing a query is done using the query() function of the mysqli class. Let’s create simple query and save it as a string.



The query() function accepts a query in the form of a string, that is why we stored our query as such. And now we execute it.

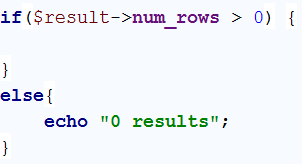


The execution of the query returns some result, so we store it in a variable. But we cannot be sure that the query was executes successfully. We need to check if the result is valid. And if it’s not we need to print an error message.

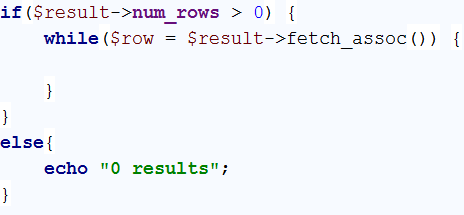


## Processing the Returned Result

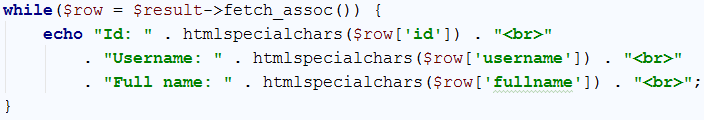
The next step is printing the results we’ve received as a result from the execution of the query.



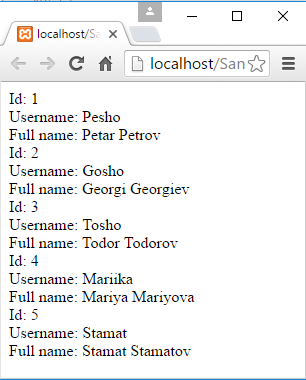
In case there are no rows, we just print “0 results”. In case there are some rows, we must print the data stored in them.



We are going to **fetch** rows, in the form of associative arrays, from the first till the last row found. This is done by calling the fetch\_assoc() function. Now that we have every row as an associative array, we can extract the data from it and print it.



In our database, our users have ids, usernames, passwords and full names. Printing the passwords would not be very good, so we’ll skip them. If you’ve done everything correctly, you should see the following result.



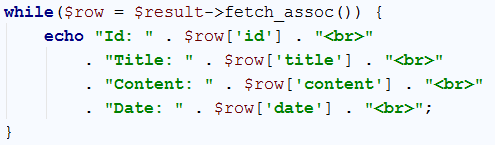
## Extracting the Posts, using mysqli

We’ve seen how to extract and print all the users from our database. Let us do the same with the posts.

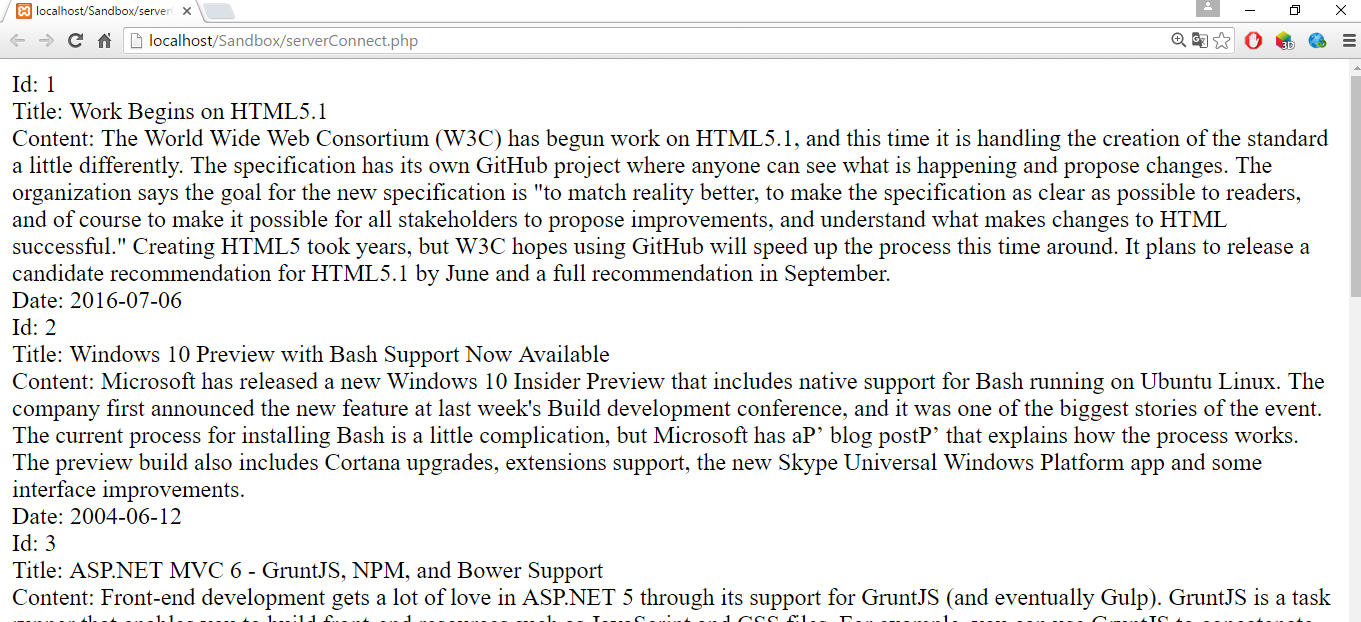
Change the query to select everything from the posts table instead of the users table.



Keep all other actions the same. The only thing you need to change, aside from the query, is the way you present the data – within the while loop that prints all the rows that have been extracted as a result from the query.



Run the script, and, if you’ve followed all steps correctly, you should see this.



## Preparing Queries with the prepare() Function.

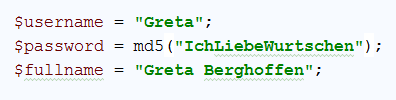
Aside from the query() function, processing queries can be done, first by preparing a query, applying parameters to it, and then executing it. Those operations can be done with the prepare() function.



The prepare function creates a query and creates **placeholders** where you have put the **question marks –** “?”. Those placeholders are later replaced with actual values using the function bind\_param().

## Constructing the Query

Let’s initialize 3 variables to hold our data about the username, password and fullname, so that we can pass them as parameters to the placeholders in the query.



Same as the other users in the database, the password is encoded with the MD5 hash. Now that we have all the data, needed to create a user in our database, we can pass those values to the placeholders with the bind\_param() function.



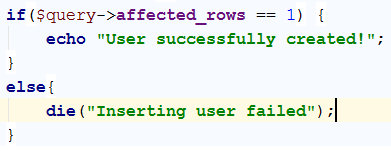
Generally, it is possible to skip specifying the data types that the placeholders will hold, but it is better to do it. There are mainly 3 types – string = s, integer = i, double = d. Setting the data types of the placeholders is done by applying a **string** **before** all other parameters in the bind\_param() function. The string should consist of the characters that were specified above. Each of those characters **corresponds** to a **placeholder**. The order of the placeholders is **zero-based**, in other words – the **first character** in the string corresponds to the **first placeholder**, and so on.

## Executing the Ready Query

Now that our query is ready, we can execute it.



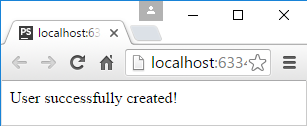
After executing the query, however, we must check if it has succeeded or failed. That is done by doing a simple check on the affected, from the query, rows.

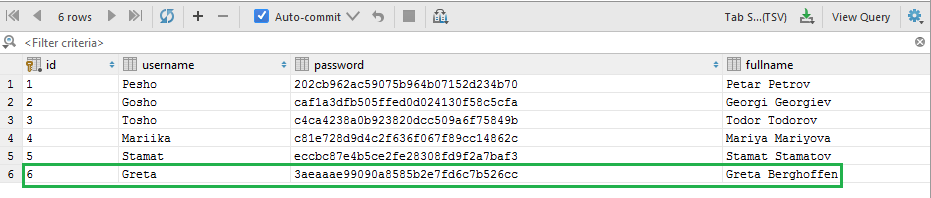


In our case, the **affected rows** should be **1**, because we are inserting **1 user** in the database. There are queries that affect **more than 1 row**, and **insert more than 1 element** to the database tables, which means that the affected rows should be more, if you are using this method. In order for the query to be completely **successful**, you should have **exactly as much affected rows**, as the **rows you intend to insert** into a certain table of the database.

## Revision

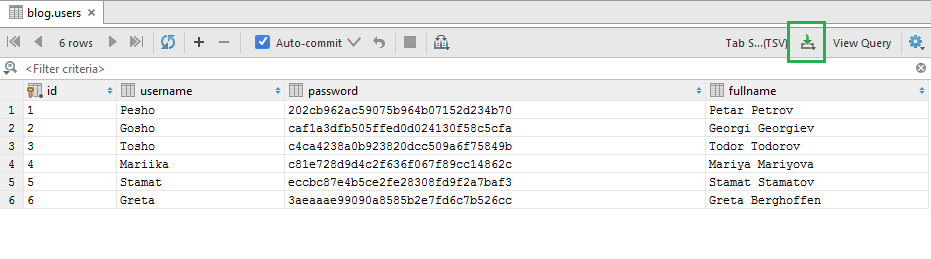
Run the current PHP script and if you’ve done everything correctly, you should see the following:

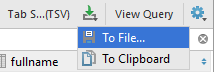




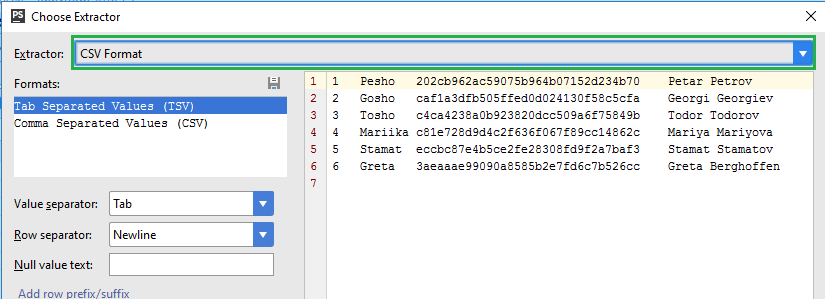
## Creating a Backup of the Database

As we said before, working with queries might be dangerous, so it would be best for us to create a backup for our database. Follow these steps to export your database tables into sql files.

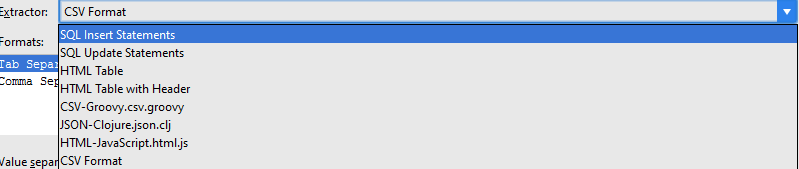




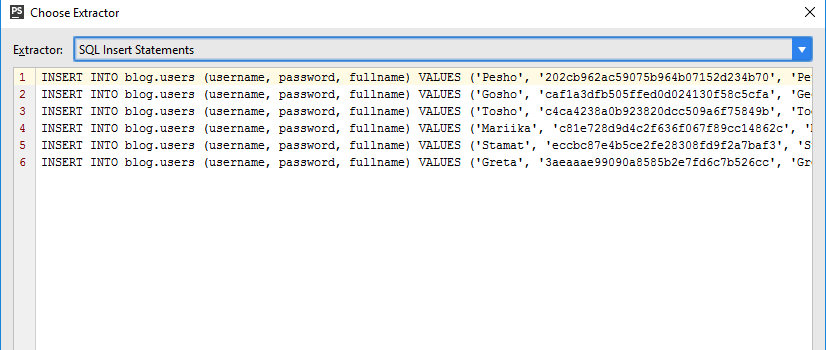
We must choose a format into which our file should be extracted.



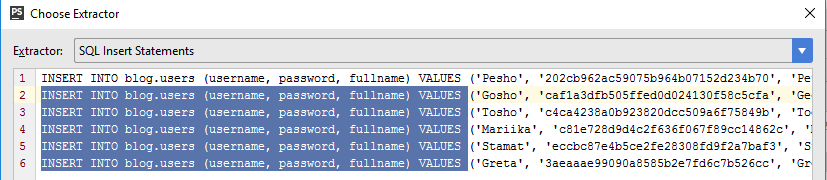
Choose “**SQL Insert Statements**”.

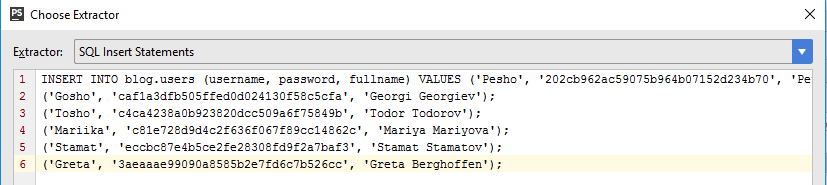


You should see the following before you:

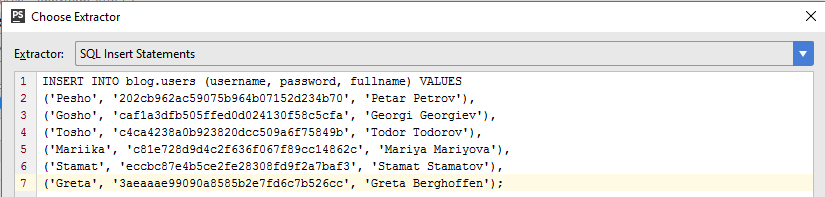


Now, delete the leading INSERT statements and values statements of all rows, besides the first one, leaving only the given values.

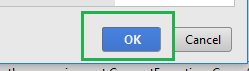




And change the closing semicolon of each row but the last one to just a comma, so that we are counting different values, one by one.



After you are done with the export format, click the OK button.



You will be shown an interface which will help you choose where to save your file and how it will be named. And with this you are ready.

If you’ve done everything correctly, you should have created a .sql file containing the extracted, into one INSERT statement, values, which we will later insert into our database, when we are restoring it. You should be familiar with the process of importing data from sql files by now, as it was specified above.

Do the same steps for the posts table.

## Working with mysqli

You’ve seen how mysqli works, and that it is nothing that hard. The SQL query syntax is the same. Try to perform, the same query actions as the ones you performed with MySQL Workbench, but this time in PHP scripts. Print the data with echo, to see if your queries work.