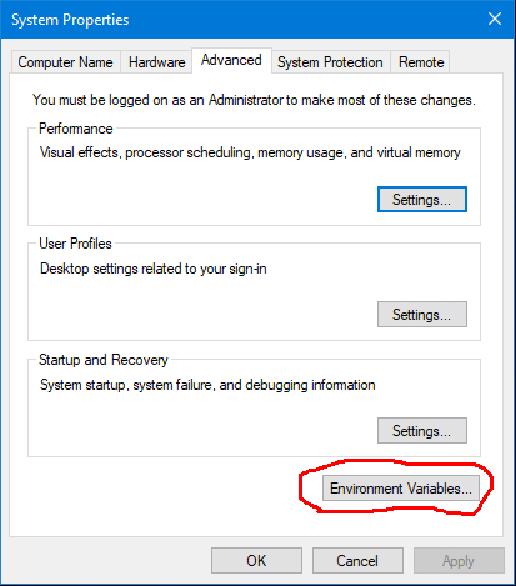
# Homework: OOP MySQL Introduction

This document defines the homework assignments for ["PHP Web Development Basic" Course @ Software University](https://softuni.bg/trainings/1746/php-web-developmentbasics-september-2017). Please submit ……………………………………………….???

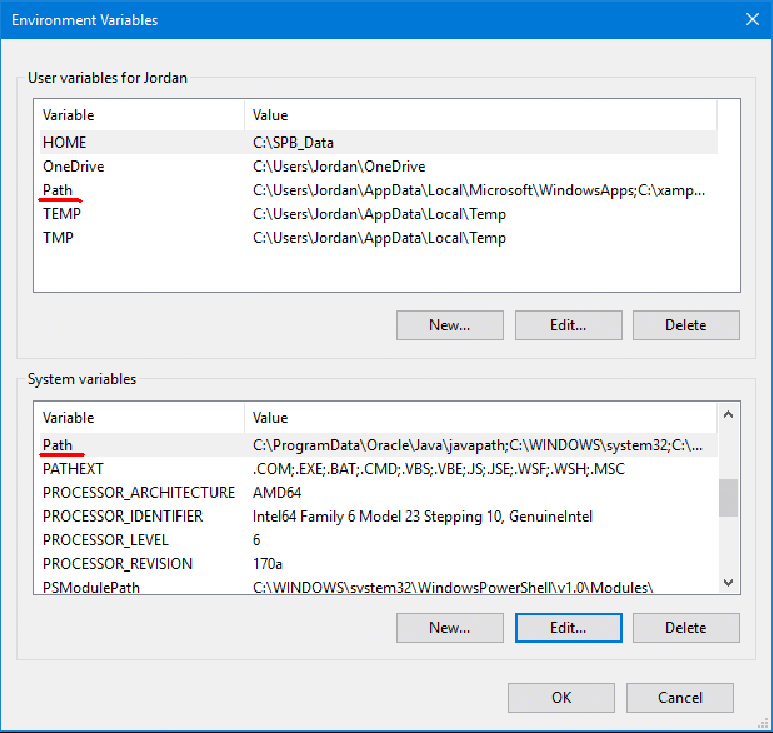
# Part I. Command Line MySQL Client

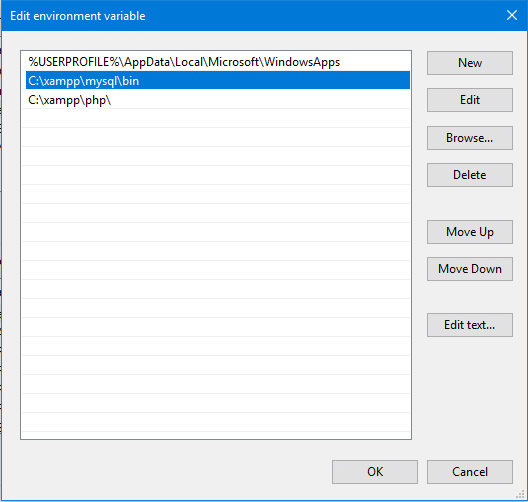
First setup your command line client for MySQL which is included in your XAMP or other installation. If you use Windows 10 then go the main menu and search **"Edit the system Environment Variables"**.

Go to **"Environment Variables"** as you can see it on the screenshot with red:



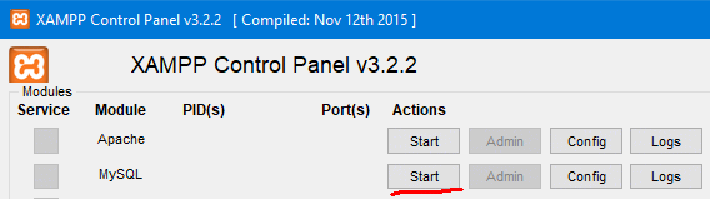
In the Environment Variables find **"Path" for your system and for your particular user** as shown on the next screenshot.



Edit Path and add **"C:\xampp\mysql\bin"** if you use XAMPP and it is installed in **C:\xampp** or find the folder where your MySQL client is if it is installed as a standalone application:

Do this for your **current user** and for the **system.**

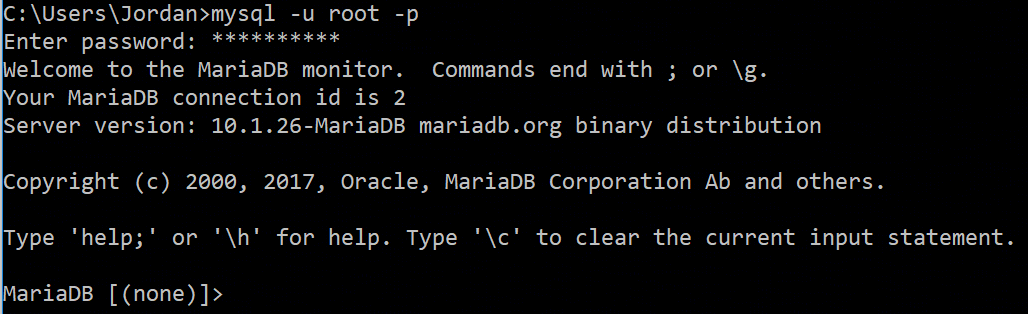
Now is the time you start your MySQL server if it is not started:



After you are sure that MySQL is started , you can connect to it through the command line like this:

**mysql –u youruser –p**

**Important! If you use MysSQL for first time it comes in XAMPP with an empty password**



Now you can write SQL queries to the server.

# 1.1. Change Your Password

1.1. First thing to do is change your password immediately! To do that we use SQL:

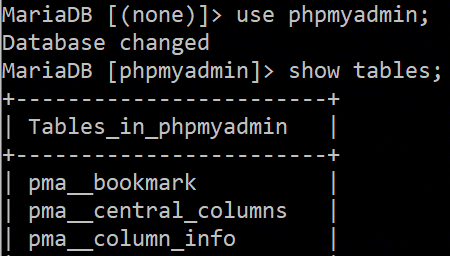
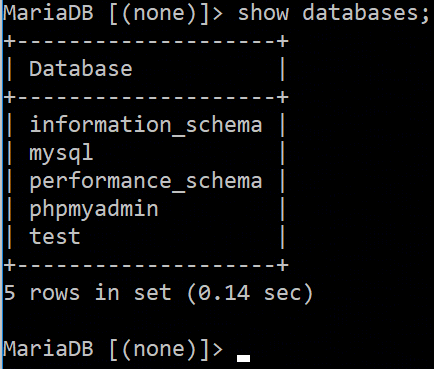
UPDATE mysql.user SET Password=PASSWORD('123myNewPass~!@#') WHERE User='root';

FLUSH PRIVILEGES;

For now you don't need to understand all SQL syntax! We are simply updating the system table **user here with the new password.**

# Browse Databases and Tables

Type in the client **show databases;** and you will see a list of all databases like:



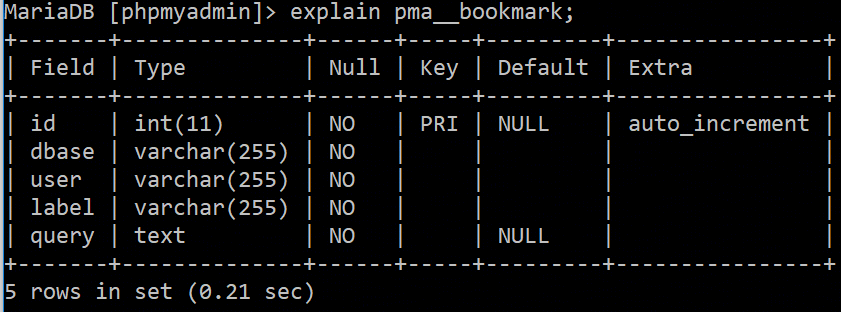
To use a particular database we write: use phpmyadmin; and then we can show tables with show tables;

On the screenshot you see only some of the tables which the web application phpMyaAdmin use to store its data.

Do this with the databases: **mysql** and **test** and **browse the tables** inside

# Explain a Table

Let's say we want to know the data structure of the table pma\_\_bookmark. Then we write: explain pma\_\_bookmark;



You get information about the **Field** (its name), **Type** (of data), can it be Null, is it a **primary key** and is there an **extra like auto\_increment.**

# Show Table's data

Now change the database to the system mysql database by: use mysql;

To show the data in a table we have to write a very simple query which uses select like this:

select \* from user;

This simply means that we select all rows from the table 'user'. The output is too large and not very good for the eyes. We will see all user data in our MySQL instance now.

To limit this query for a particular user and field (password hash) it will be:

**select Password from user where user.User = 'root';**

|  |
| --- |
| **Important Notice!** On a production server only the system administrator and an authorized programmer should ever do changes to the database! Usually changes are introduced by scripts after testing, not by hand! |

Using the MySQL client can be weary some for large administrative tasks because there might be a lot of writing. It is **useful** when you have only **initial command line access to the server** and you want to set things up. You may set up a remote connection to MySQL and then use a graphic client like HeidiSQL, Datagrip, MySQL Workbench or other.

From this point on we will use HeidiSQL

# Part II. Solving Particular Problems

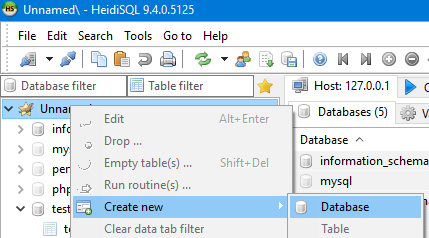
From this point on we will use **HeidiSQL (**[**https://www.heidisql.com/**](https://www.heidisql.com/)**)** and this you will see on the screenshots. For more help on the software go to the forum: <https://www.heidisql.com/forum.php>

# Problem 1. Simple Phone Book

Our task is to create a **new database phonebook** which will contain a single table **phones** with the following fields:

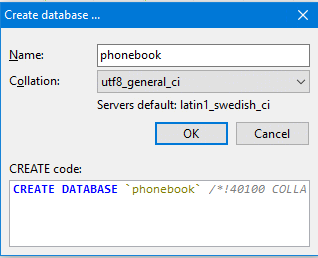
**person\_id** – integer and auto increment value for every row and a primary key. **first\_name** – a varchar up to 255 symbols. **last\_name** – a varchar up to 255 symbols. **numbe**r – it should be start with the country code like 359123456789. Choose a convenient type

# Step 1. Create the Database



On the left click on the main element (Unnamed) with the **right mouse button.** Then **"Create new"** and **"Database".** Choose a **name phonebook** and a **collation (utf\_general\_ci).** You can see this on the next screenshot.

HeidySQL will always **show you the query** which it executes! Everything you do with the **mouse** results in an **SQL query** which you could write also in PHP, mysql command line client or any other client to MySQL.

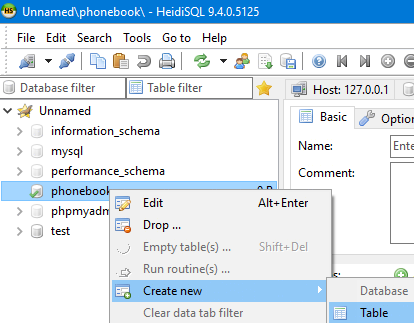
Press OK.

If you could go back to your command line client and type: **show databases;** you would see now the new database in the list.

Let's go to the next step and create the table.

# Step 2. Create the Table Phones

**Important Notice!** Some MySQL servers don't work well with camel case table names. You might be in a situation transferring a whole database from one server to another where the importing server imports everything with small letters and then your SQL queries can break!

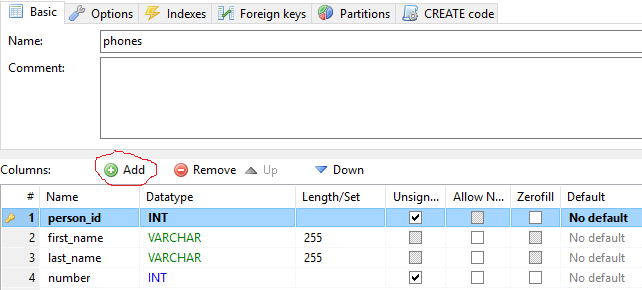
Creating a table is analog to creating a database. See the screenshot.

First you give your table a name **phones.** Then you must add the fields (or columns) in your table by clicking on the **green plus and Add** the field one by one.

Be sure to use the proper **Datatype of the field. For the person\_id** use the **right mouse button -> Create New Index -> Primary.**

The result is something similar to the screenshot below.

To create the new table press **"Save".** You can see the SQL that is output for each of your actions.

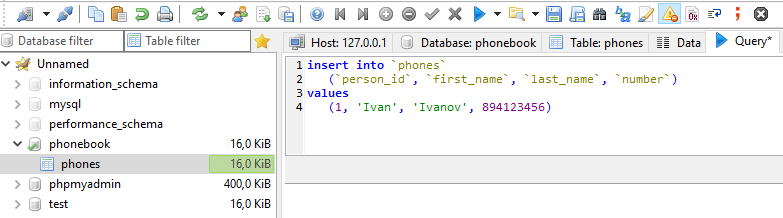


When you created with the graphic tool the table on the server was executed the following:

|  |
| --- |
| **CREATE** **TABLE** `phones` (  `person\_id` **INT** **UNSIGNED** **NOT** **NULL**,  `first\_name` **VARCHAR**(255) **NOT** **NULL**,  `last\_name` **VARCHAR**(255) **NOT** **NULL**,  `number` **INT** **UNSIGNED** **NOT** **NULL**,  **PRIMARY** **KEY** (`person\_id`)  )  **COLLATE**='utf8\_general\_ci'  **ENGINE**=**InnoDB**  ; |

# Problem 2. Insert a Person and Phone

A phonebook does nothing if you don't use it. So now you have to insert your name and phone into the phonebook. In HeidiSQL click on your database (phonebook) and then on Query. Our query will be this:



|  |
| --- |
| **insert** **into** `phonebook`  (`person\_id`, `first\_name`, `last\_name`, `number`)  **values**  (**1**, 'Ivan', 'Ivanov', 894123456) |

**Notice!** Back apostrophes **( ` )** are used in MySQL to be sure that the table name or field (column name) is not a system keyword of MySQL ( like: insert into `insert` … ) It is convenient not to use system keywords as names! To be sure a good practice is always to put back apostrophes.

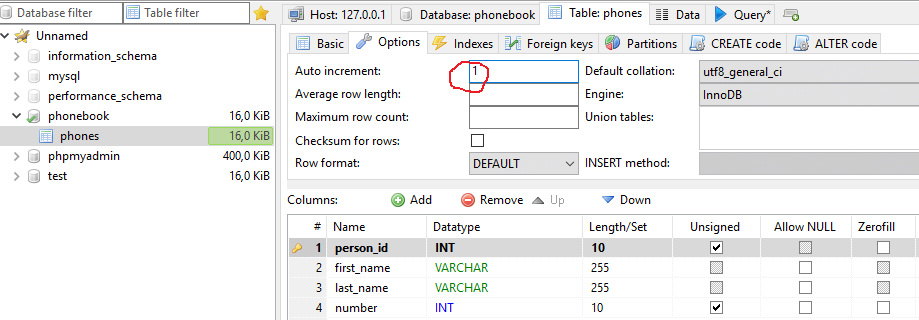
To run the query press F9. You can add much more people but there are two problems to be solved:

* You can't use a number like 359894122456 because it brings an **out of range error**
* You have **to fill the person\_id by hand**.

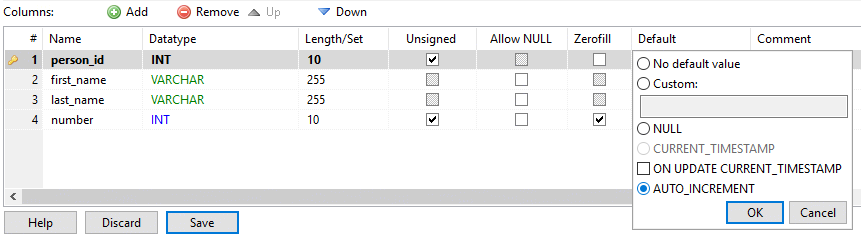
# Problem 3. Redefine Table Structure

Redefine your table structure by **changing the datatype of number** to a more convenient. Add an **auto\_increment** to **person\_id** so that you might pass **null** instead of a number. This way MySQL does that for you.

To set a value as auto\_increment we need to go to options and set the offset:



Then the next step is to change the default value of **person\_id** to auto increment like this:



Press OK and Save to do the job. Note what HeidiSQL did automatically:

|  |
| --- |
| **ALTER** **TABLE** `phones` **ALTER** `number` **DROP** **DEFAULT**; |
| **ALTER** **TABLE** `phones` **AUTO\_INCREMENT**=1,  **CHANGE** **COLUMN** `person\_id` `person\_id` **INT**(10) **UNSIGNED** **NOT** **NULL** **AUTO\_INCREMENT** **FIRST**,  **CHANGE** **COLUMN** `number` `number` **INT**(10) **UNSIGNED** **ZEROFILL** **NOT** **NULL** **AFTER** `last\_name`;  **SELECT** `DEFAULT\_COLLATION\_NAME` **FROM** `information\_schema`.`SCHEMATA` **WHERE** `SCHEMA\_NAME`='phonebook'; |

ALTER TABLE changes the **structure** of your table!

Now test what you have done with a new query like this:

|  |
| --- |
| **insert** **into** `phonebook`  (`person\_id`, `first\_name`, `last\_name`, `number`)  **values**  ( **null**, 'Ivan', 'Stoyanov', 359894123456) |

Enter three persons with the first name being **Ivan**!

# Problem 4. All Phone Numbers of Ivan

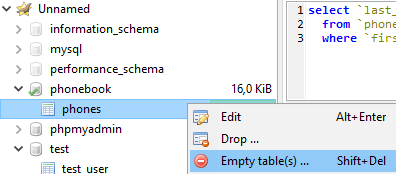
Write a simple query that will get the last name and phone number of all persons who have Ivan as their first name.

|  |
| --- |
| **select** `last\_name`, `number`  **from** `phones`  **where** `first\_name` = 'Ivan' |

Execute this query in **HeidiSQL** and from the **command line client**. You should get the same output!

# Problem 5. Truncate Table Phones

Image that we collected too much garbage in the table and we don't need any data inside. We can delete all data without deleting the table in HeidSQL by a right mouse on **phones and**:



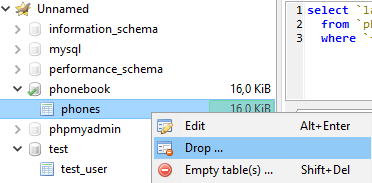
The corresponding code is:

|  |
| --- |
| **TRUNCATE** `phones`; |

This will not change the structure of phones but will empty it by all rows of information!

# Problem 6. Delete a Whole Table

Image that we don't need phones anymore. Then we can drop the whole table and delete it all. It is gone forever!

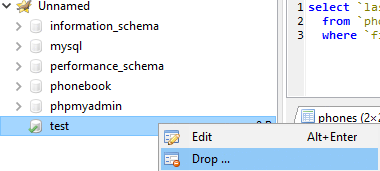


The corresponding SQL is:

|  |
| --- |
| **DROP** **TABLE** `phones`; |

# Problem 7. Deleting a Whole Databse

Image we don't need the database called **test.** We need to drop it like this:



The corresponding SQL is:

|  |
| --- |
| **DROP** **DATABASE** `test`; |

Drop the test table in XAMPP and see what happened in the console client by typing: show databases;

**Notice! It is a convenient practice to backup before you do truncate or drop! Your data will be deleted and there is no plan B!**