MySQL Notes, Flask

Introduction

- MySQL slides provides basic and advanced concepts of MySQL. Its designed for beginners and professionals.
- MySQL is a relational database management system. It is opensource and free.
- Includes all topics of MySQL database such as insert record, update record, delete record, select record, create table, drop table etc. There are also given MySQL interview questions to help you better understand the MySQL database.

- Before learning MySQL, you must have the basic knowledge of computer fundamentals.

What is MySQL

- MySQL is a fast, easy to use relational database. It is currently the most popular open-source database. It is very commonly used in conjunction with PHP scripts to create powerful and dynamic server-side applications.

- MySQL is used for many small and big businesses. It is developed, marketed and supported by MySQL AB, a Swedish company
- With MySQL you can store any records in your company, today companies require databases to store huge data coming in and out on a daily basis

Why MySQL

- MySQL is an open-source database so you don't have to pay a single penny to use it.
- MySQL is a very powerful program so it can handle a large set of functionality of the most expensive and powerful database packages.
- MySQL is customizable because it is an open source database and the open-source GPL license facilitates programmers to modify the SQL software according to their own specific environment.
- MySQL is quicker than other databases so it can work well even with the large data set.
- MySQL supports many operating systems with many languages like PHP, PERL, C, C++, JAVA, etc.
- MySQL uses a standard form of the well-known SQL data language.
- MySQL is very friendly with PHP, the most popular language for web development.

- MySQL supports large databases, up to 50 million rows or more in a table. The default file size limit for a table is 4GB, but you can increase this (if your operating system can handle it) to a theoretical limit of 8 million terabytes (TB).

MySQL Features

- Easy to use: MySQL is easy to use. You have to get only the basic knowledge of SQL. You can build and interact with MySQL with only a few simple SQL statements.
- It is secure: MySQL consist of a solid data security layer that protects sensitive data from intruders. Passwords are encrypted in MySQL.
- Client/ Server Architecture: MySQL follows a client /server architecture. There is a database server (MySQL) and arbitrarily many clients (application programs), which communicate with the server; that is, they query data, save changes, etc.

- Free to download: MySQL is free to use and you can download it from MySQL official website.
- It is scalable: MySQL can handle almost any amount of data, up to as much as 50 million rows or more. Check out more on https://www.mysql.com/

From https://www.mysql.com/

Here are some of MySQL users

YouTube PayPal Google facebook

ebay cisco LinkedIn

UBER TESLA NETFLIX GitHub

MODCOM Institute of Technology

MySQL Data Types

- A Data Type specifies a particular type of data, like integer.
- As tutorial points puts it ;

"Properly defining the fields in a table is important to the overall optimization of your database. You should use only the type and size of field you really need to use".

- Check out more on https://www.tutorialspoint.com/mysql/mysqldata-types.htm

MODCOM Institute of Technology

Install MySQL

- We will use XAMPP server this lesson, XAMPP is an easy to install Apache distribution containing MariaDB, PHP, and Perl. Just download and start the installer. It's that easy.

- Download it here https://www.apachefriends.org/download.html
- You can also use wamp server here ; http://www.wampserver.com/en/
- You can also install from http://www.mysql.com

MODCOM Institute of Technology

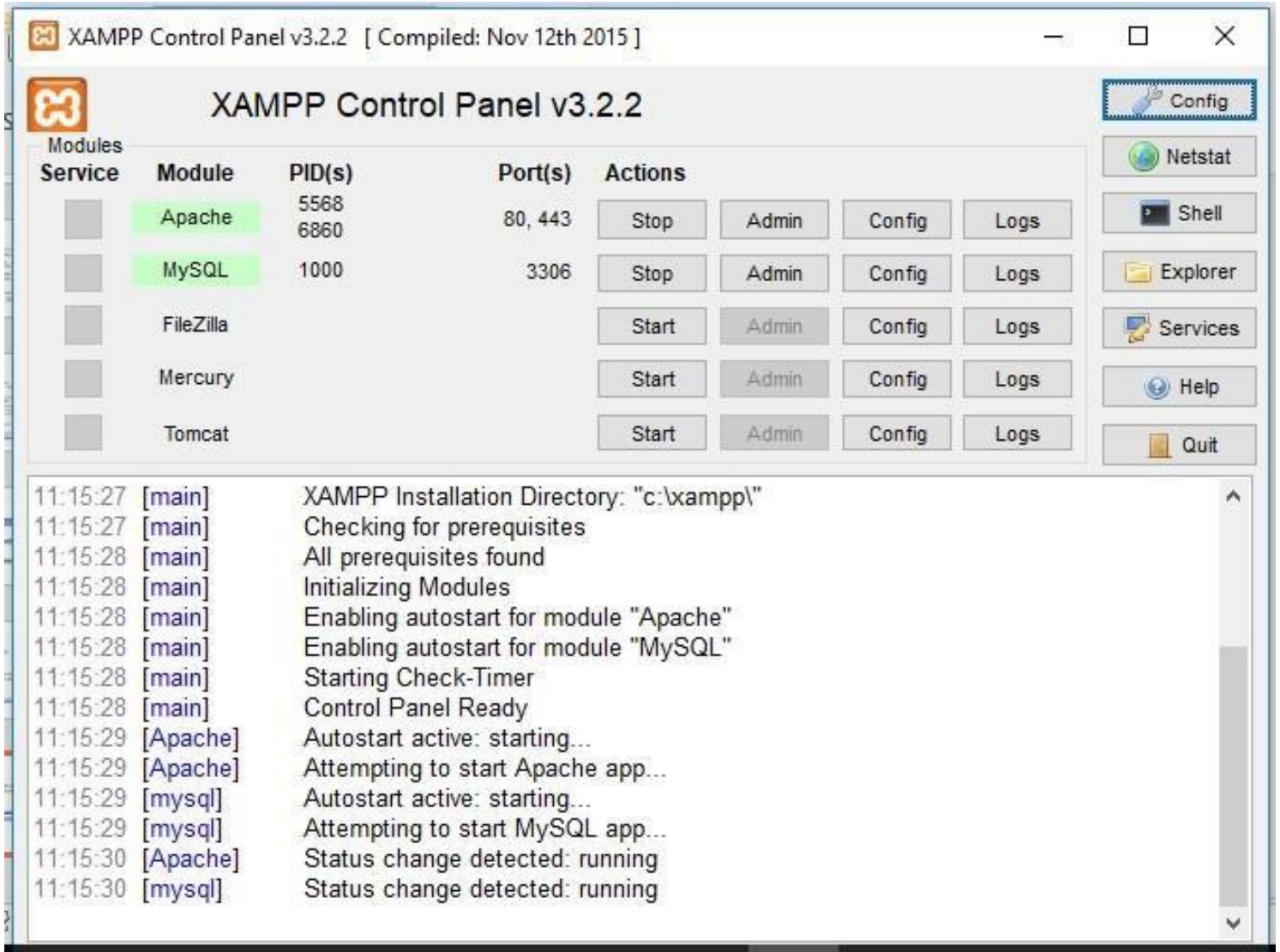
Start Xampp Server.

Start Xampp from the Search, then start Apache, Mysql, they should appear in green.

Open any browser and type

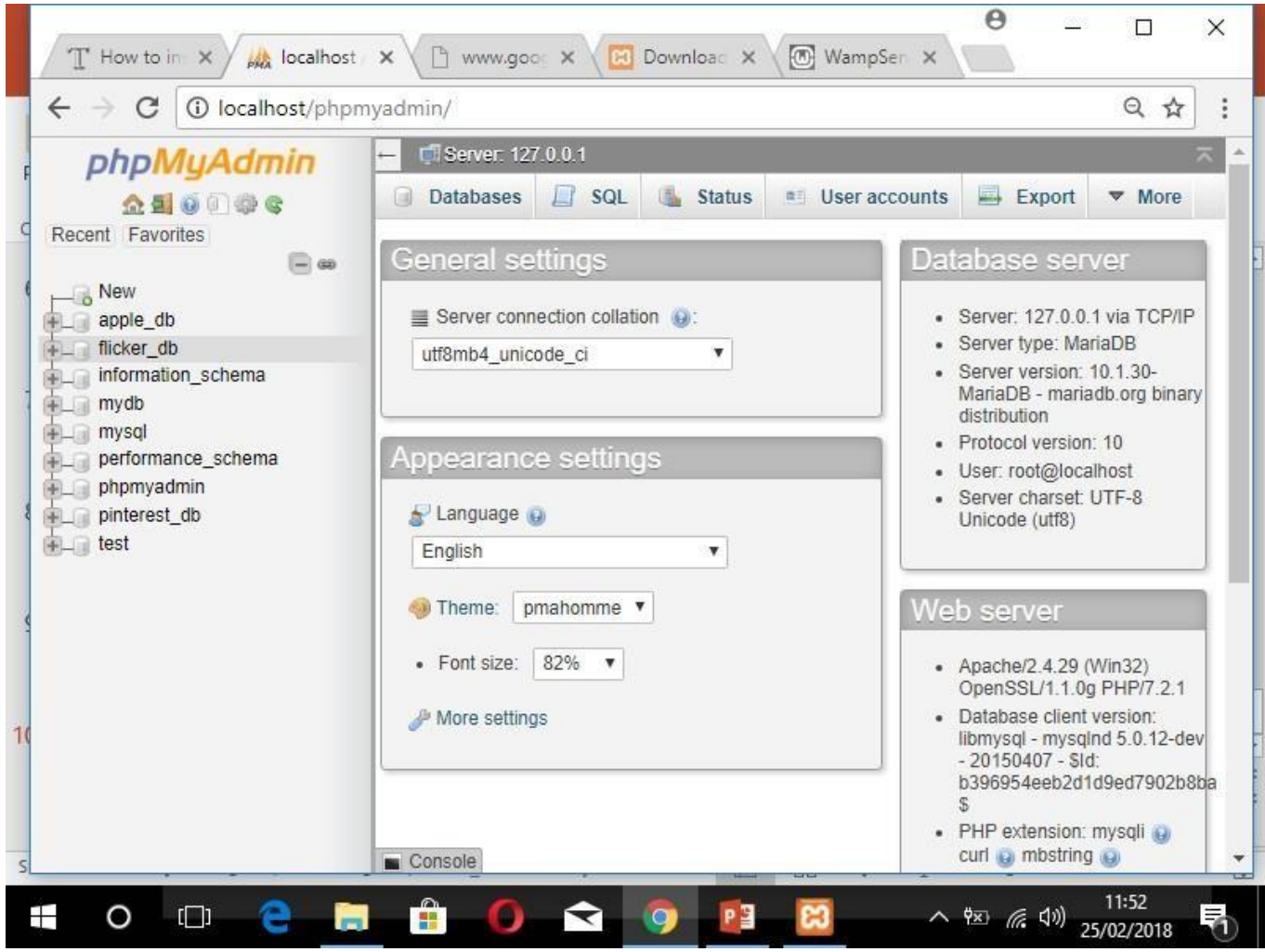
localhost/phpmyadmin

The next screen opens



Welcome to phpmyadmin

With phpmyadmin, you can create databases, Tables and manipulate data



CASE STUDY

- Uhai Hospital is a medium sized hospital located in western parts in Nairobi, the hospital serves patients from different parts of the country, The hospital uses manual methods to manage its employee, in our MySQL lesson we will automating serves at Uhai hospital by creating a patients registration system.

- Now, we first create a database for Uhai Hospital.

- A database stores data in categories called tables, for this example we can have tables such as , patients, doctors, medicines, payments, staff, suppliers etc, etc

patients

patient_id VARCHAR(20) dob VARCHAR (20) fname TEXT (20) lname TEXT (20) gender TEXT (20) reg_date VARCHAR(20)

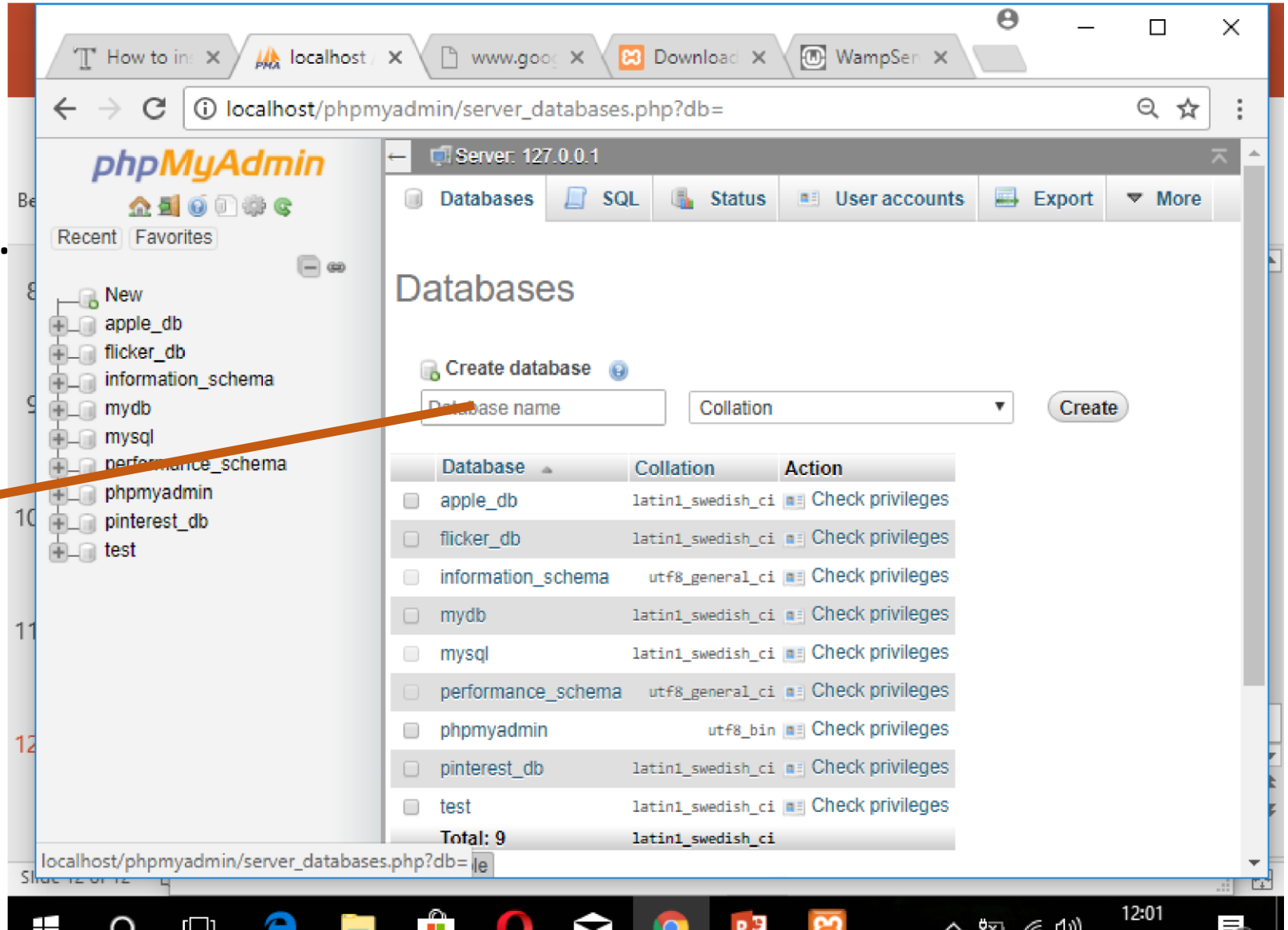
medicines

drug_id VARCHAR(20) name VARCHAR (20) brand TEXT (20) exp_date TEXT (20) reg_date TEXT (20)

Creating A Database

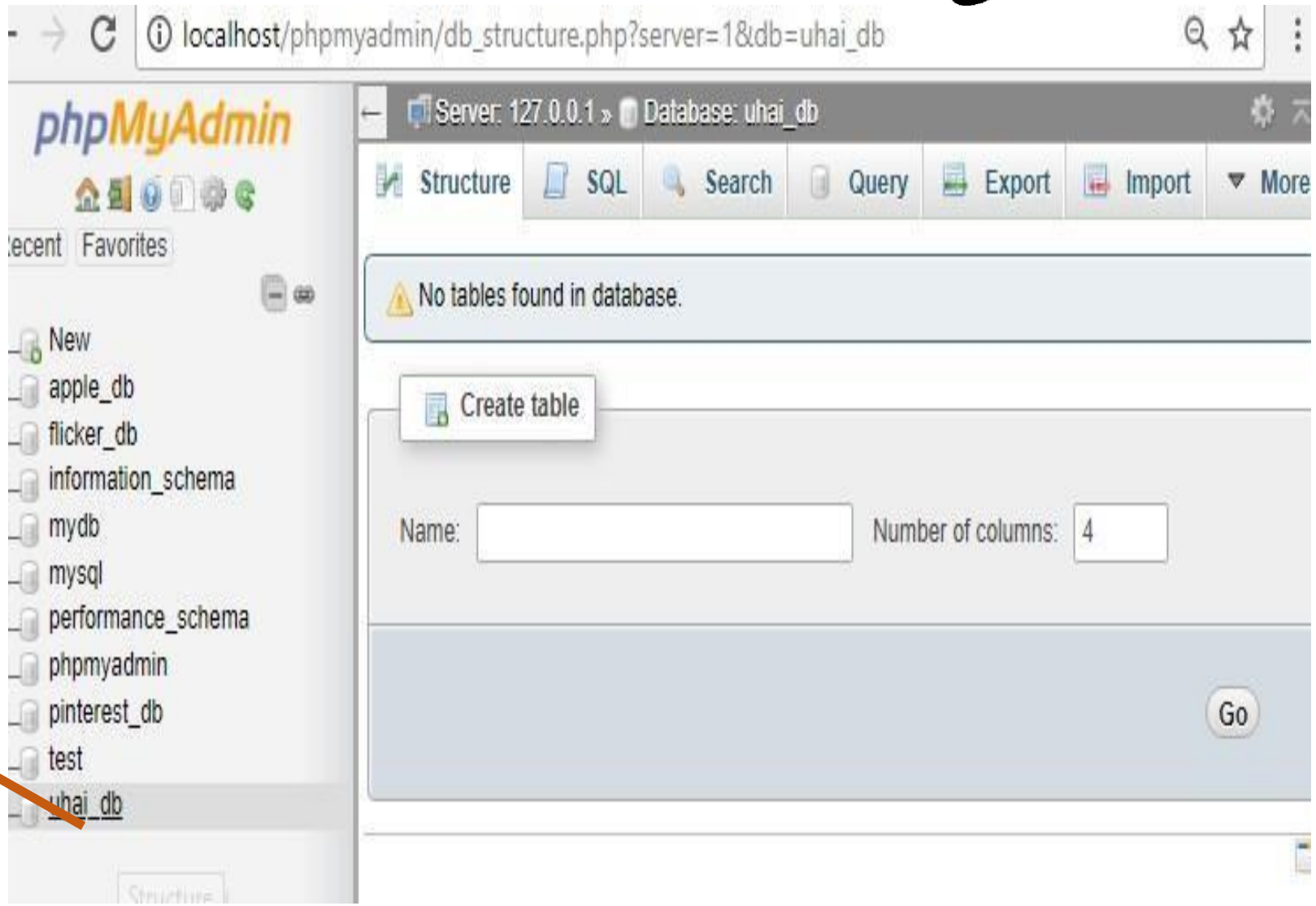
Go to phpmyadmin and click on databases.

Enter database name then click on ‘**Create**’



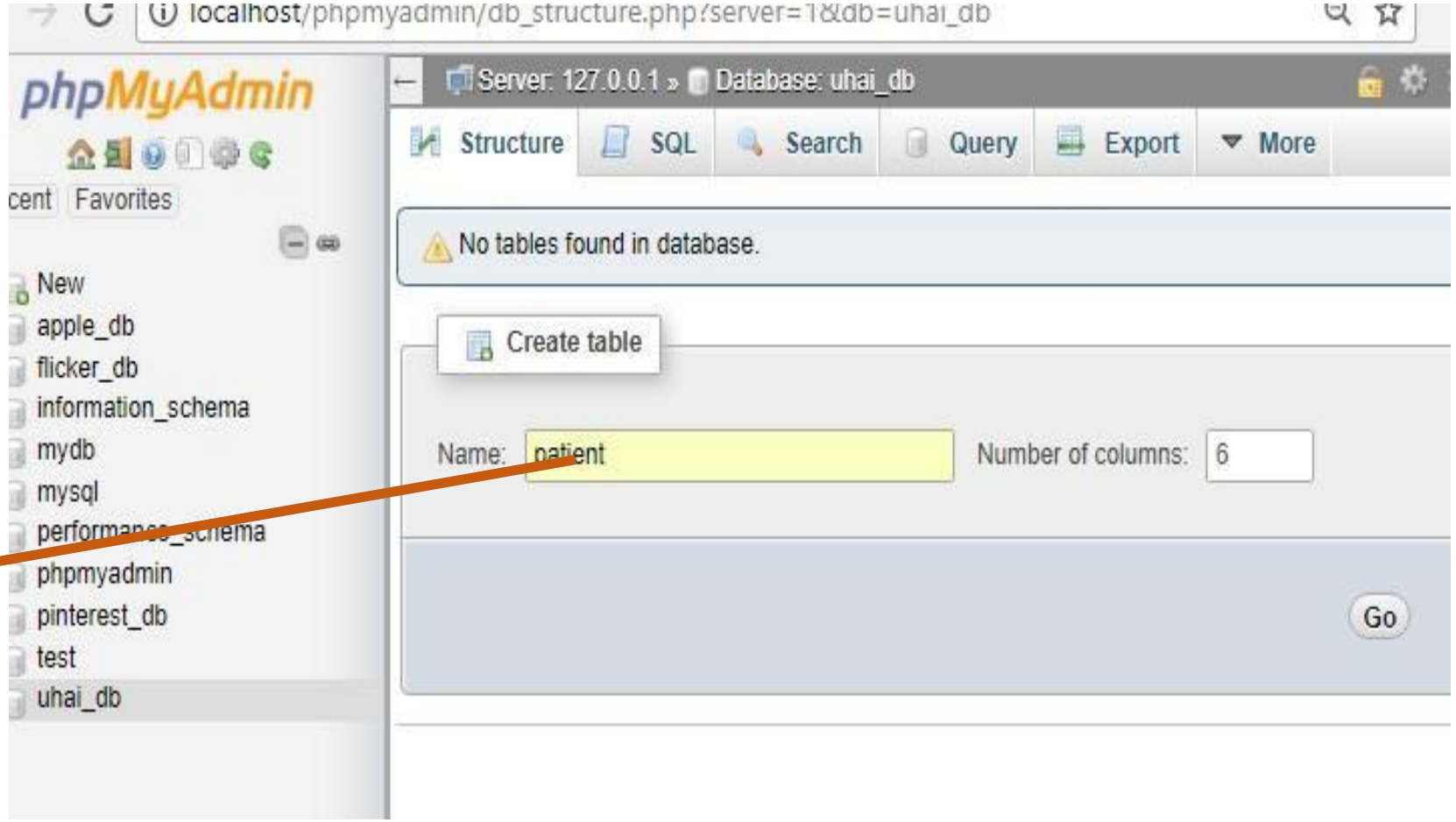
Database created.

Created database appears on the LEFT



Creating A Table

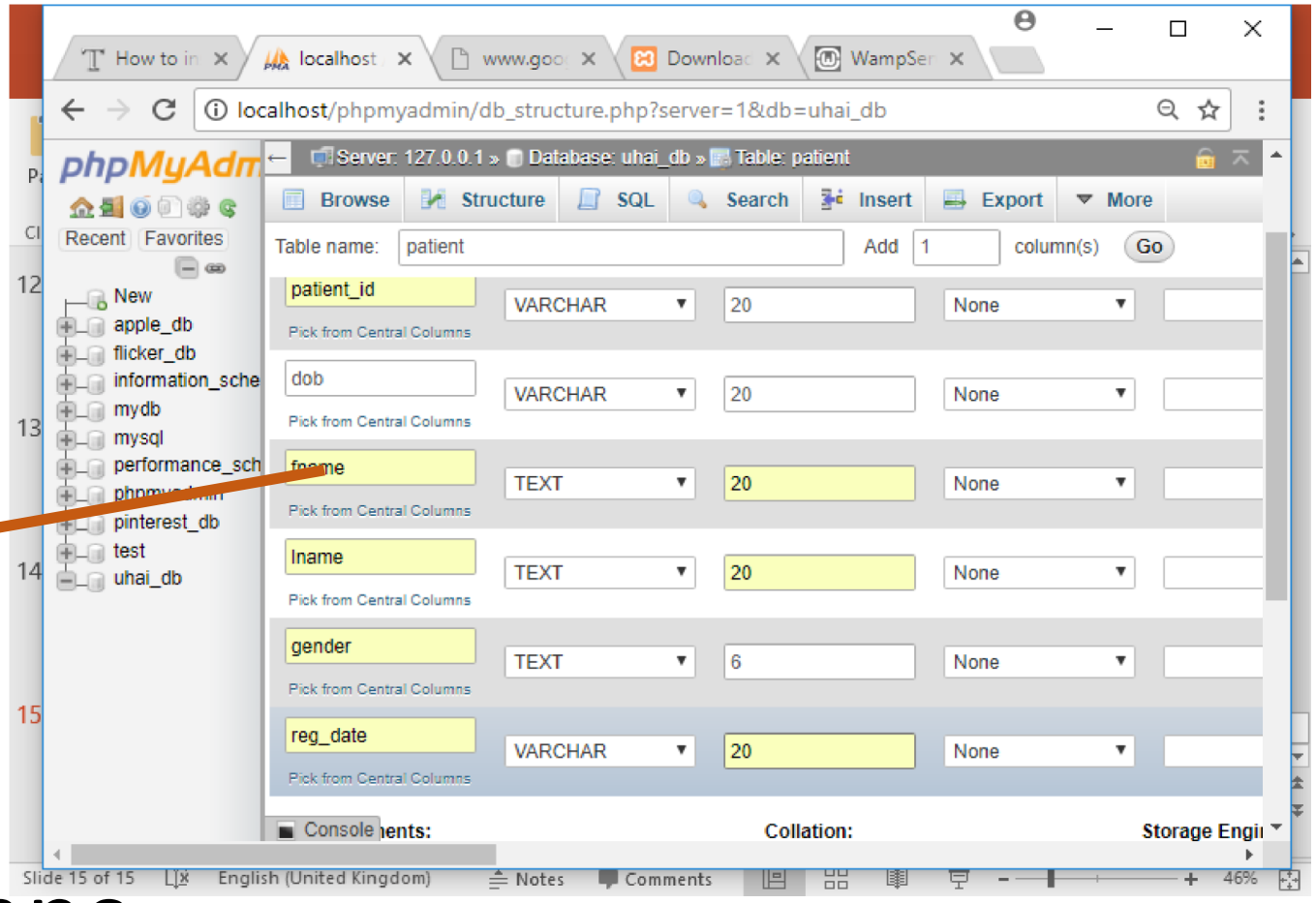
Creating a patients table, while your 'uhai_db' is selected.



- Enter table name '**patients**' and **6 columns,** click **Go**

Next, you will be presented with a page to enter the **columns, data type** and **length**

- Enter columns as shown , **type and length**, click on **Go**



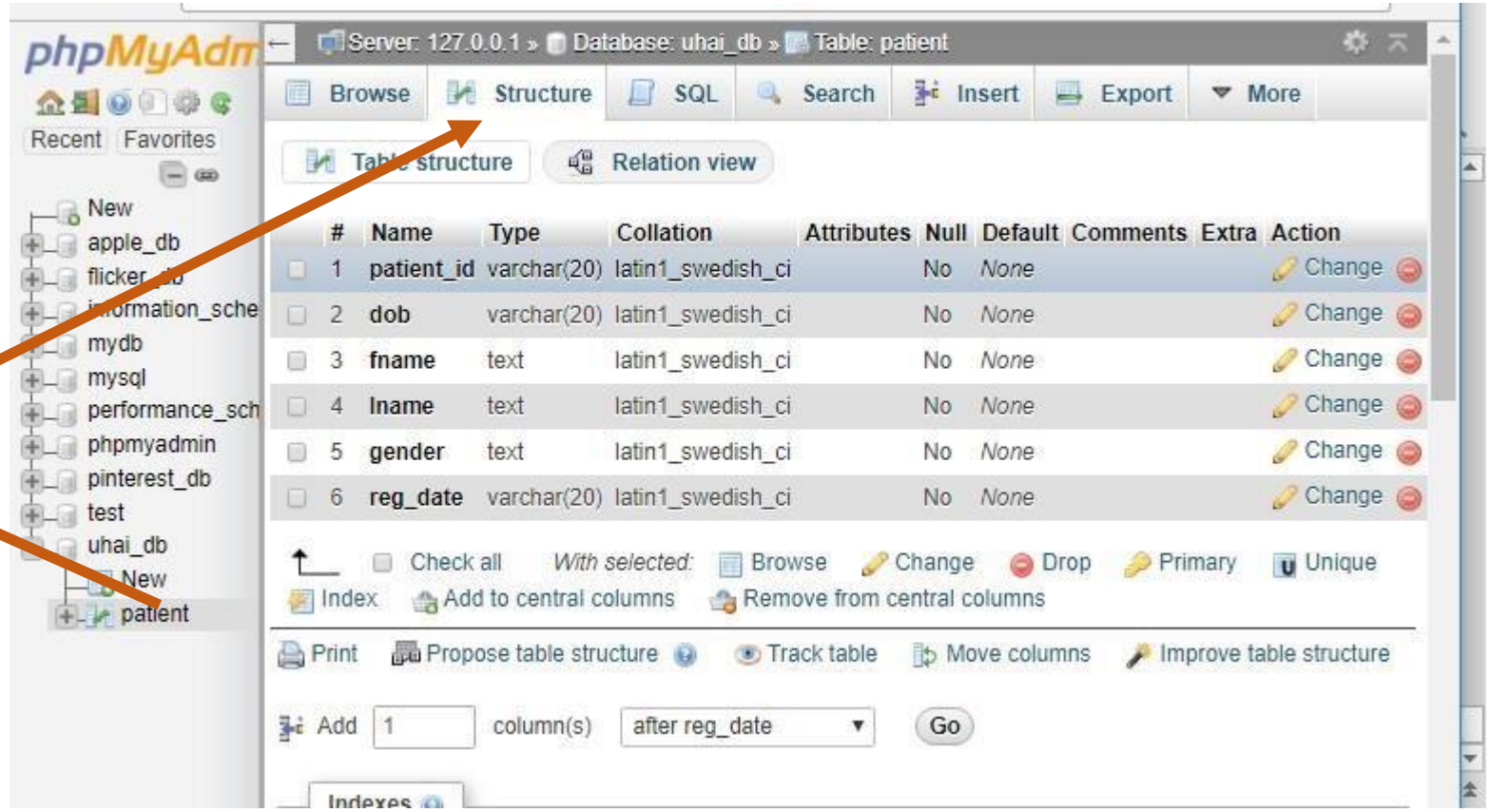
Creating Table Columns

Table Created.

The table has been created.

When you are selected on any table, you can

Browse, view table structure, view SQL, export data etc

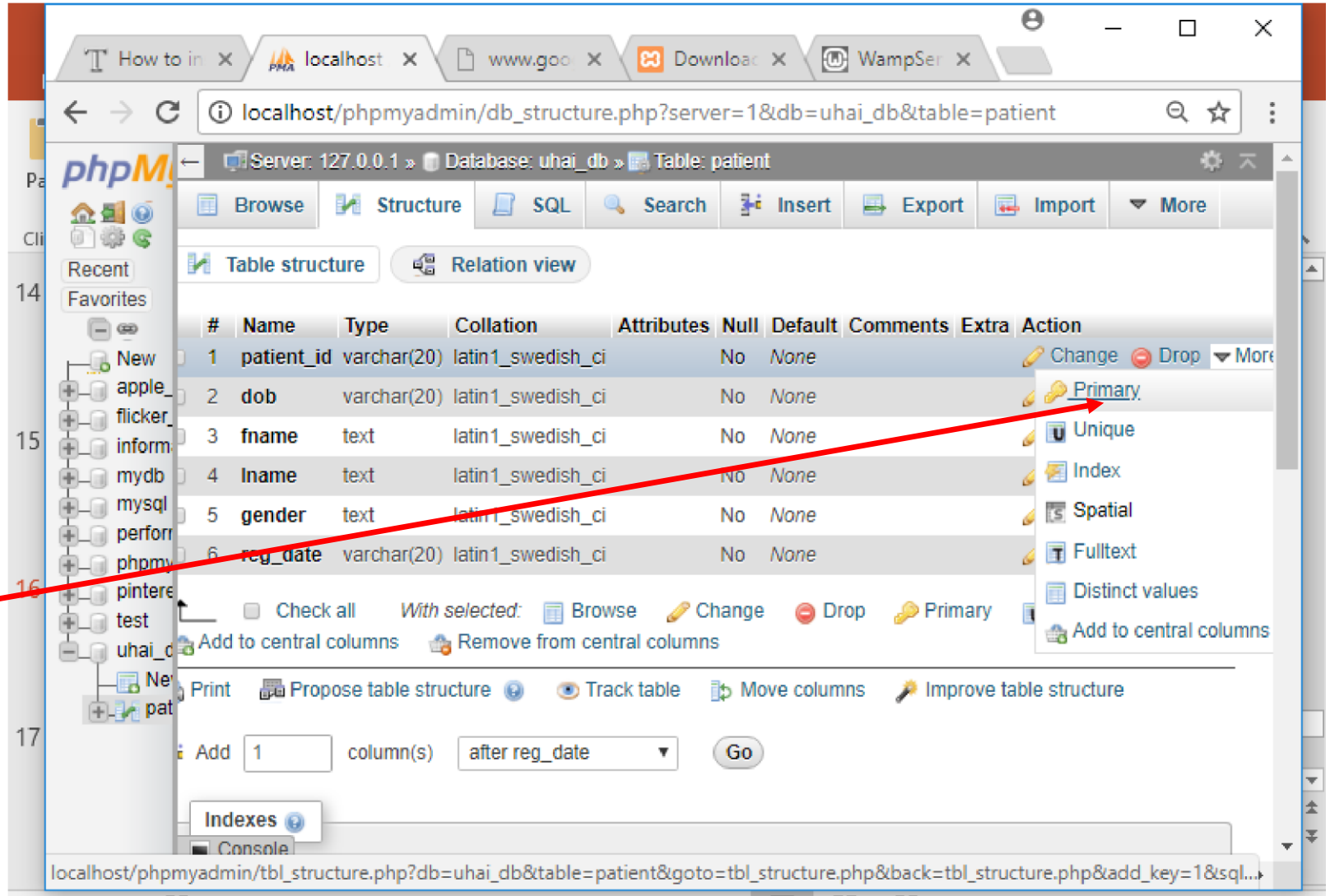


Each table MUST have a Primary Key, which is unique column to identify each row uniquely.

In this case patient_id in our Primary Key.

In your table structure, click on primary in the patient_id column. ALTER TABLE ? Click **OK**

Primary Key cannot be repeated.



Database and Tables

- In the previous slides, we created a database, a table and columns.

- We also set a Primary Key to patient_id to make it a unique column to indent each patient individually.

- Next we will learn SQL – Sequential Query Language , we will work with patients table created previously in this slides.

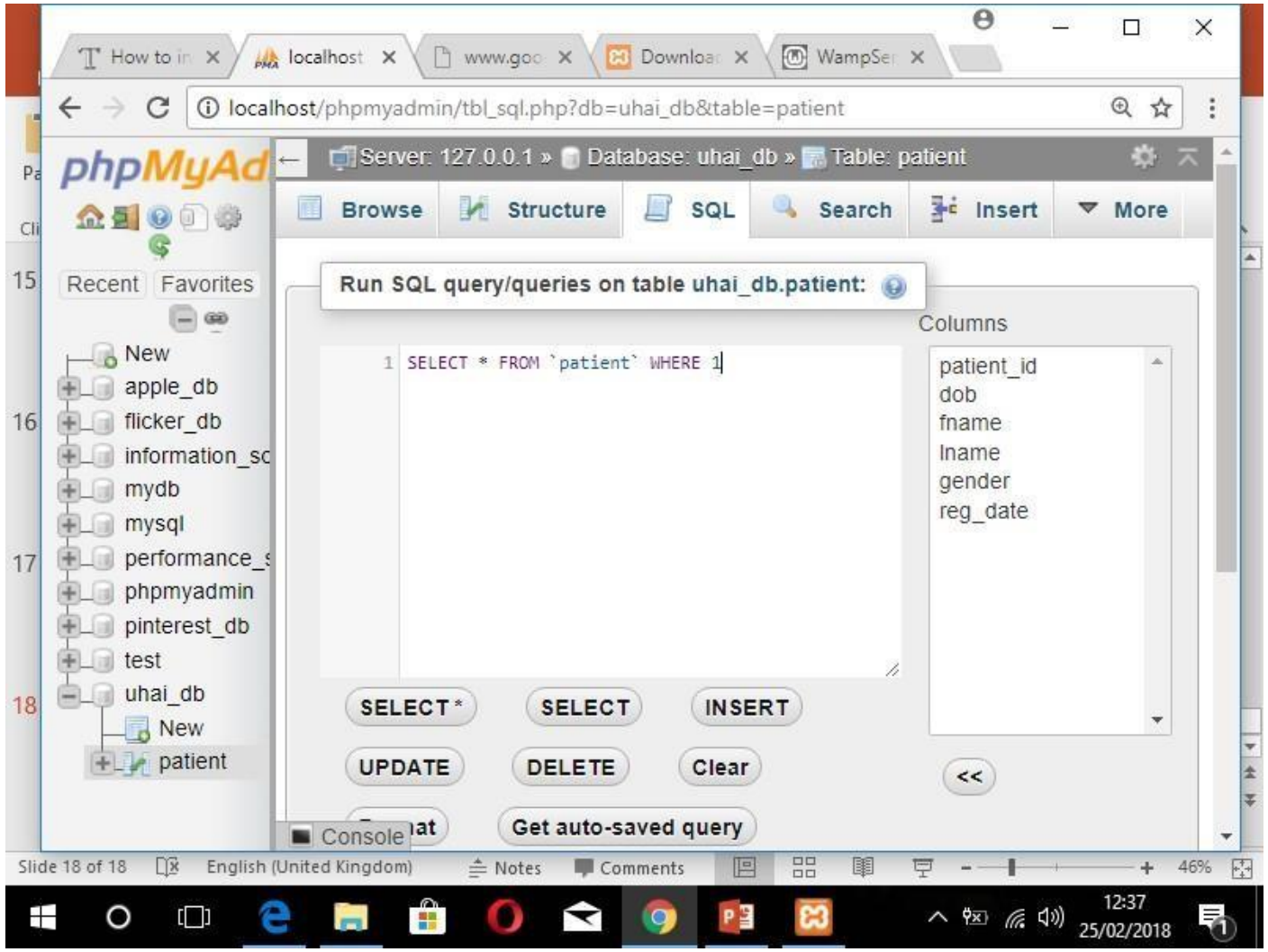
- While in PHPMYADMIN ‘’ Click on database, then ‘patients’ table.

- At the top bar menu select SQL. See next.

SQL (*Structured Query Language*) is used to perform operations on the records stored in database such as updating records, deleting records, creating and modifying tables, views etc.

SQL is just a query language, it is not a database. To perform SQL queries, you need to install any database for example Oracle, MySQL, MongoDB, PostGre SQL, SQL Server, DB2 etc.

We use SQL in this CASE.



We will look at the following SQL Keywords:

- INSERT • INTO
- SELECT
- FROM
- WHERE
- BETWEEN
- ORDER BY
- LIMIT
- UPDATE
- DELETE
- GROUP BY
- AND
- ETC

INSERT INTO

Used to insert records into a table

MODCOM
Institute of Technology

Syntax

- Below are rules for **INSERT** query.

INSERT INTO table_name (*all columns separated by commas*) VALUES (*all values as per columns, separated by commas, single quoted individually*)

Example

- Insert into **patients** table

```
INSERT INTO patient(patient_id, dob, fname, lname, gender, reg_date) VALUES ('50002','20100405','Anita','Albert','F','2018-5-5')
```

Its NOT mandatory to single quote the table and columns

This query will insert a record directly into patients table

NB: In SQL, table name and columns name can be quoted with tick marks ``, but its not a MUST, in this slide the table names and columns names are not quoted with tick marks

SELECT - FROM

Used to select records from a table

MODCOM
Institute of Technology

Syntax

- Below are rules for **SELECT** query.

```
SELECT columns separated by commas FROM
table_name;
```

Syntax

- Below are rules for **SELECT** query. **Patient** is our table name

```
//shows 2 colms , dob, fname
SELECT dob, fname FROM patient;
//shows 3 colms , dob, fname, lname
SELECT dob, fname, lname FROM patient;
```

```
//shows 1 colms , patient id
SELECT patient_id FROM patient;
//shows all columns
SELECT *  FROM patient;
```

WHERE

Selects specific records from the table based on given columns

Syntax

- Below are rules for **WHERE Clause** query.

```
SELECT columns separated by commas FROM table_name
WHERE column = value
```

Syntax

- Below are rules for **WHERE Clause** query.

```
//returns all patients who have gender F
SELECT * FROM patient WHERE gender = 'F'
```

```
//returns all patients who have dob 2010-4-4
SELECT * FROM patient WHERE dob = '2010-4-4'
//returns all patients who have patient_id  20003
SELECT * FROM patient WHERE patient_id = '20003'
```

Syntax

- SELECT by any 2 columns. i.e select a patient who was born on 2010-44 and is male

```
//returns all patients who have gender F
SELECT * FROM patient WHERE dob = '2010-4-4' AND
gender = 'M'
```

```
//NB: if there no match, returns zero rows
```

Syntax

- SELECT patient who were born between 1999-5-5 AND 2004-4-4

```
SELECT * FROM patient WHERE dob  BETWEEN '19995-5'
AND '2004-4-4'
```

```
//NB: if there no match, returns zero rows
```

ORDER BY

Orders selected records by either ascending or descending

Syntax

- SELECT all patients and ORDER BY dob ascending or descending

```
//Shows patients displayed with they dob ascending
SELECT * FROM patient ORDER BY dob ASC

//Shows patients displayed with they dob descending
SELECT * FROM patient ORDER BY dob DSC
```

LIMIT

Limits the records displayed with a specific number

Syntax

- SELECT all patients , LIMIT to 10 ONLY

- //Shows patients displayed, cuts down to 10

```
SELECT * FROM patient LIMIT 10
```

//Shows patients displayed with they dob descending limit only to 4 records

```
SELECT * FROM patient ORDER BY dob DESC LIMIT 4
```

DELETE

Deletes records from the table

Syntax

- DELETE

```
•//delete all records
DELETE FROM patient
```

```
//deletes all patients with gender male
DELETE FROM patient WHERE gender = 'M'
//deletes a patient with id 10008
DELETE FROM patient WHERE patient_id = '10008'
```

UPDATE

Updates an existing record

Syntax

- Below are rules for UPDATE query.

UPDATE table_name SET colm=value WHERE colm = value In UPDATE, we need to state the colm we are updating based on another colm, if it exists, see example next

Example of a update query

```
UPDATE patient SET dob='2010-2-2' WHERE patient_id = '10008'
```

Saving Data Using Flask to MySQL

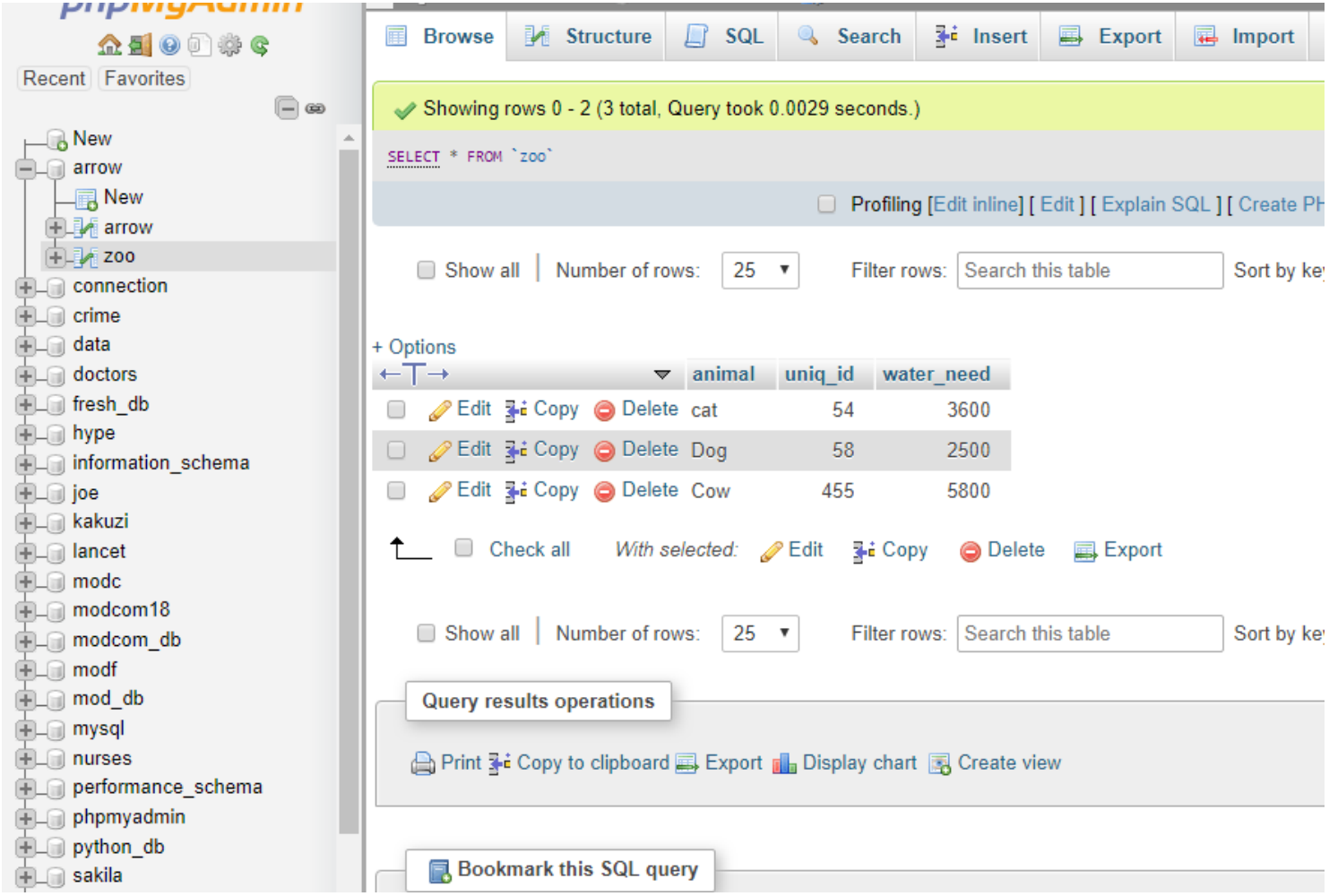
Consider we want to save data in a database named ‘**arrow’** , table named ‘**zoo’** with collumns below;

1. animal, uniq_id, water_need

   First we need to do a Flask template named **add.html in** templates folder Add this HTML form **<code>**

   In the form below all inputs have names attributes, the values are *animal, uniq_id,water_need*

Db name: **arrow**. Table name: **zoo**



Add.html

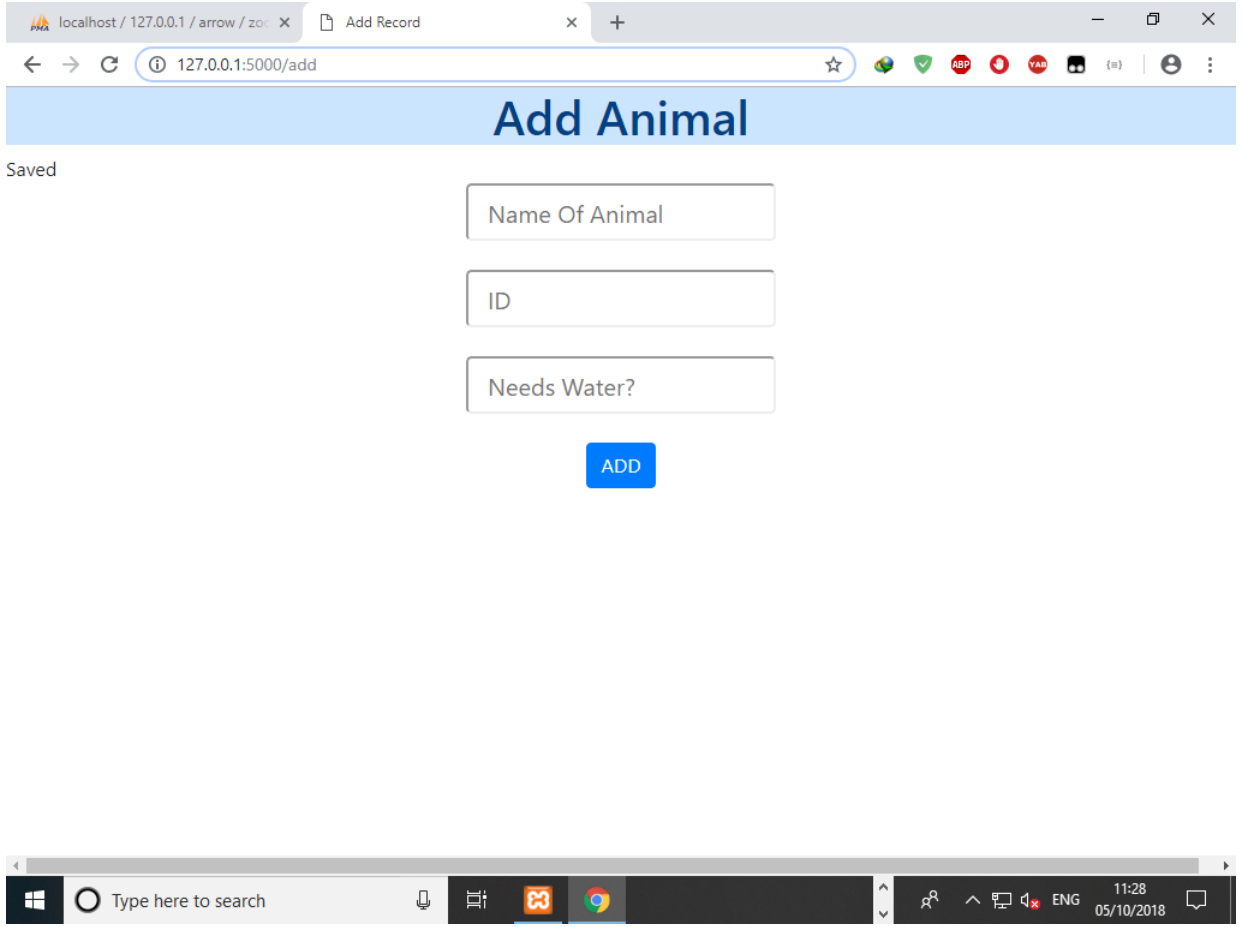
```html
<form action="/add" method="post">
    <input name="animal" type="text"
     placeholder="Name of animal">
     <br><br>

    <input name="uniq_id" type="text"
     placeholder="id">
<br><br>
    <input name="water_need" type="text" class="form-control-lg text-capitalize"
     placeholder="needs water?"><br><br>
    <input type="submit" value="add" >
</form>
```

Then in Flask, we need ot do a route to handle the posted data, Note that the form uses POST method, read more on POST/GET here;

https://www.w3schools.com/tags/att_form_method.asp

Your form should look like below table



in your app.py create a route named /add

add the blow code to save data to a table named ‘zoo’

```
from flask import request
import pymysql
@app.route("/add", methods=['POST','GET'])
def add():
    if request.method == 'POST':
        animal = request.form['animal']
        uniq_id = request.form['uniq_id']
        water_need = request.form['water_need']

        con = pymysql.connect("localhost","root", "", "arrow")
        cursor = con.cursor()
        sql = "INSERT INTO `zoo`(`animal`, `uniq_id`, `water_need`) VALUES (%s,%s,%s)"
```

```
        try:
            cursor.execute(sql,(animal,uniq_id,water_need))
            con.commit()
            return render_template('add.html', msg = "Saved")
        except:
            con.rollback()
            return render_template('add.html', msg = "Failed")
    else:
        return render_template('add.html')
```

On the browser;

http://127.0.0.1:5000/add

Explanation

import pymysql **# this is needed for us to coonect our database in localhost/phpmyadmin**

@app.route("**/add**", methods=['**POST**','**GET**']) # this explains that only 2 methods are allowed POST, GET

Below code check if user posted some inputs from the form

```
if request.method == 'POST':
    animal = request.form['animal']
    uniq_id = request.form['uniq_id']
    water_need = request.form['water_need']
```

If user posted, thenm we connect to our database using pymysql

Cursor object is used to execute SQL

```
con = pymysql.connect("localhost","root", "", "arrow")
cursor = con.cursor()
```

 # prepare SQL ready to execute, the SQl insert into 'zoo' table

```
sql = "INSERT INTO `zoo`(`animal`, `uniq_id`, `water_need`) VALUES (%s,%s,%s)"
```

 # Note that the SQL does not have values ot save, we have ony passes %s, 5s represent the values parsed in execute next lines.

 # below execute SQL, provinding the three values entered by user from the form check: animal = request.form['animal']

```
cursor.execute(sql,(animal,uniq_id,water_need))
con.commit()# once executed commit changes to table
```

Getting Records from the zoo table

Create the follwing route in your python file in Flask.

```
@app.route("/all")
def all():
    #connect the database
    con = pymysql.connect("localhost", "root", "", "arrow")
    # Create cursor to be used in execuitn gour sql
    cursor = con.cursor()

    #prepare the SQl, selecting all records in 'zoo' table
    sql = "SELECT * FROM zoo"

    # execute the SQL
    cursor.execute(sql)
```

```
    # check how many rows were returned, if its zero, return
error
    # message back to template
    if cursor.rowcount==0:
        return render_template('all.html', msg="No Records")
    else:
        # return rows found back to template
        rows = cursor.fetchall()
        return render_template('all.html', rows=rows)
```

Next, we need to bind our message or the rows the our template

Create a html file named **all.html** in templates folder

Add the following code to all.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>All Animals</title>
</head>
<body>
 <!--We bind the message, incase its returned-->
 <h1>My Animals</h1>
 <b>Nice One!</b>
<h4>{{ msg }}</h4>
 <!--We loop through all the rows returned-->
  <!--here, we put the data in heading...a table can also be used-->
 <table border="1" width="80%">
 <!--table headers-->
 <tr>
     <th>Name</th>
     <th>Uniq Id</th>
     <th>Water Need</th>
 </tr>

 {% for row in rows %}
     <tr>
         <td>{{ row[0] }}</td>
         <td>{{ row[1] }}</td>
         <td>{{ row[2] }}</td>
     </tr>
```

```
  {% endfor %}
</table>

</body>
</html>
```

We should get the dat ain a table.

On the browser:

http://127.0.0.1:5000/all

localhost / 127.0.0.1 / arrow / zoo

All Animals

127.0.0.1:5000/all

My Animals

Nice One!

Name	Uniq Id	Water Need
cat	54	3600
Dog	58	2500
Cow	455	5800

Type here to search

ENG 11:34 05/10/2018

Check this project on github

https://github.com/modcominstitute/Flask