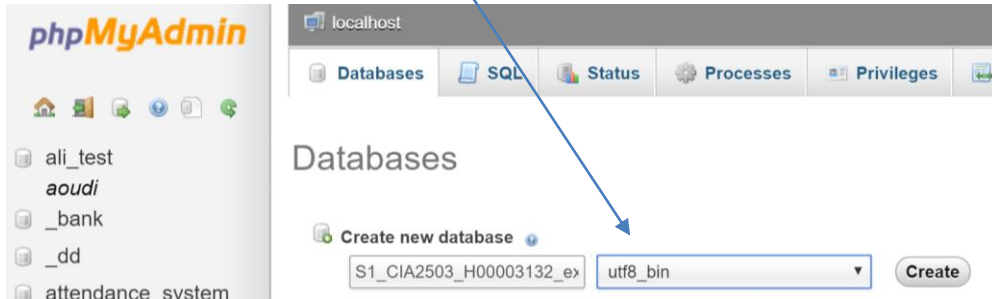


## PHP/ MYSQL viewing DB records – single table example

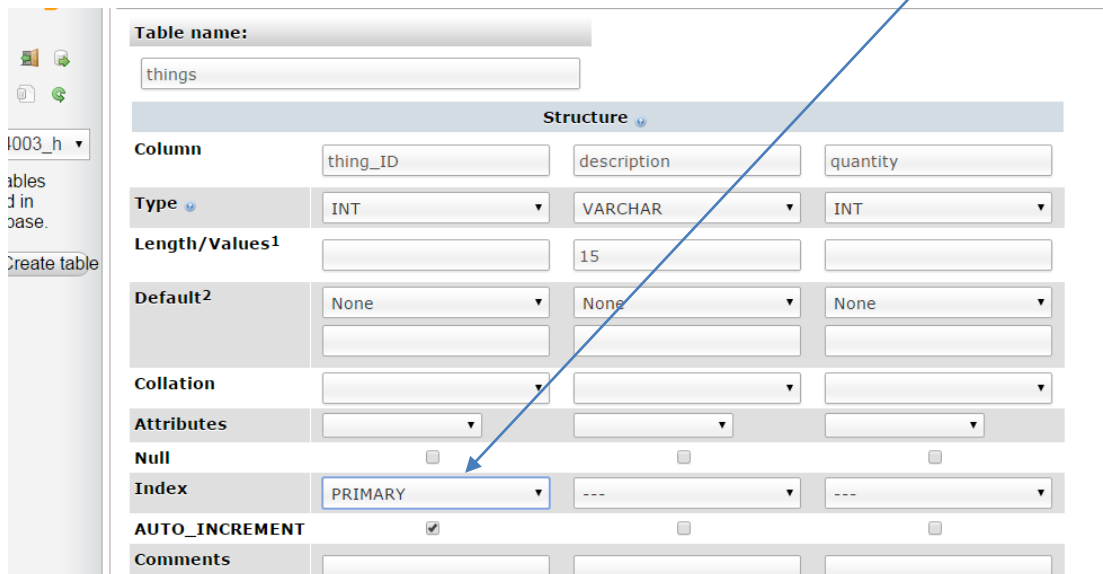
**The Backend:** create a database in MYSQL. DB's can start with Sx\_CIA2503\_H00xxxxx\_DBname where S1 = section 1, S2 = section 2, then course#, then your ID then DB name – use **utf8** to support different languages. For example:



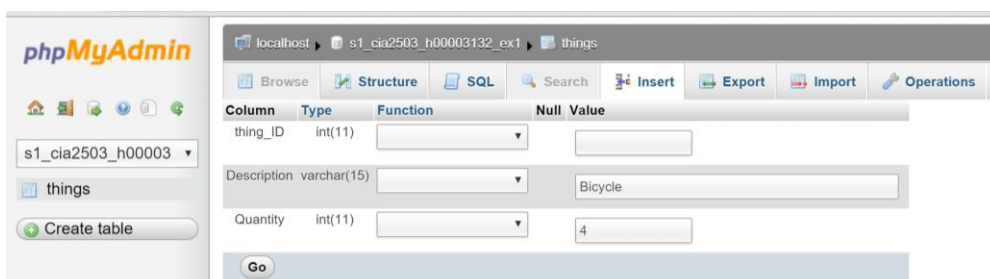
Then add tables, fields (columns) and dummy data (NB: do not add data to AI columns)



Then add **columns** (or fields). Below we have added 3 with the first being the **primary key** with **AI**



Next add **dummy data** – so that we have something to look at when we view from a web page.



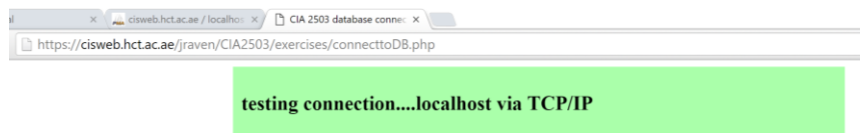
**The Front end:** Create a php page and add the database connection using the code below:

```
10 <?php
11 //create connection
12 $con = new mysqli("localhost", "student", "student", "CIB4003")
13 if ($con->connect_errno) { //failed
14     echo "Failed to connect to MySQL: (" . $con->connect_errno . "
15 }
16 echo $con->host_info . "\n"; //success
17 //connected
18 echo "connected to DB";
```

**Note:** A copy of the basic code is on G drive – you just need to change the DB name

### Third step:

Copy the doc to the cisweb server using ftp – use File Explorer or the tool built into NotePad++. Then check that it is working on the server – you have now connected to the database!



**Lesson two - view records:** First rename the connection string php file as **viewthings.php** and then alter it to display records. This needs a recordset that collects data using an SQL “SELECT \* from table” command then display the records using a while loop.

This is what the PHP needs to do:

- Connects to the Database – puts out an error message if this fails (see code above)
- Runs a SELECT SQL command that collects the data from the database and stores them in an array called \$result – checks how many records and if none outputs a message
- Otherwise – uses a while loop to output each record in a list (see code below)

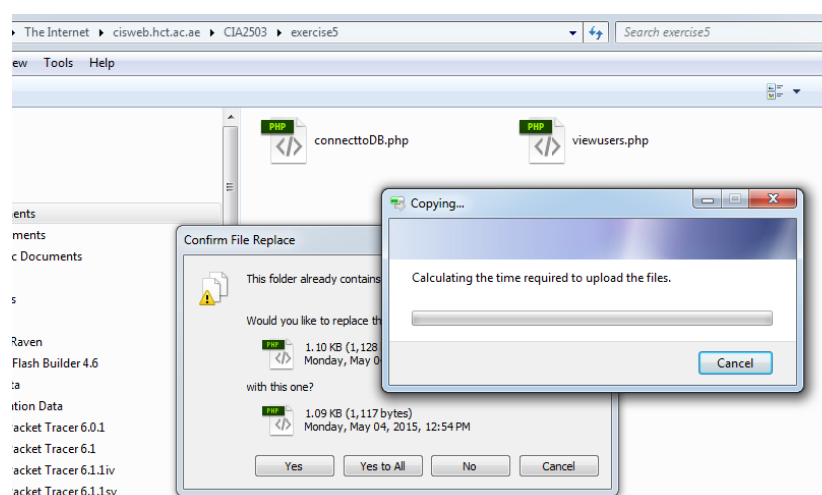
**Note:** An easy way to do this is to copy the SQL from PHPMYAdmin (no need for the LIMIT)

A screenshot of the phpMyAdmin web interface. On the left sidebar, the database 's1\_cia2503\_h00003' is selected, and the table 'things' is highlighted. The main panel shows a SQL query: 'SELECT \* FROM `things` LIMIT 0, 30'. Below the query, it says 'Showing rows 0 - 1 (~2 total)'. At the bottom, there is a table with two rows of data. The table has columns 'thing\_ID', 'Description', and 'Quantity'. The first row shows '1', 'Bicycle', and '4'. The second row shows '2', 'Surfboard', and '1'. Each row has links for 'Edit', 'Inline Edit', 'Copy', and 'Delete'.

Study the code below to understand how it works then copy it into viewthings.php.

```
echo "connected to DB";
//run SQL query
$sql = "SELECT * FROM things";
$result = mysqli_query($con,$sql) or die(mysqli_error());
//output result
if(mysqli_num_rows($result)==0) //no records found
{
    echo "<p>no records in DB".mysqli_num_rows($result)."</p>";
}
else // records were found in DB
{
    echo "<p>You have: ".mysqli_num_rows($result). " things</p>";
    ?>
    <ul>
    <?php
    while($row = mysqli_fetch_array($result)) { //loops through records
        echo "<li>Thing_ID: ".$row['thing_ID']. "</li>";
        echo "<li>Thing description: ".$row['description']. "</li>";
        echo "<li>Number: ".$row['quantity']. "</li>";
    } //end of loop
    echo "</ul>";
} //end of else
```

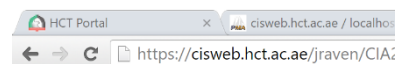
Then copy this file to the webserver and check that it works.



**Note:** An error 404 means you did not upload it or have the wrong URL. An error 500 means you have an error in your php code. If the page is completely empty then that is a PHP error also.

**Exercise:** Display all the things in your database in a list like →

**Individual activity:** change the output from a list to a table and hide the "localhost..." text. Also add a <main> and style the page using a centered box 60% wide and coloured.



## CIA 2503 exercise one

### My things

localhost via TCP/IP connected to DB

You have: 2 things

- Thing\_ID: 1
- Thing description: Bicycle
- Number: 4
- Thing\_ID: 2
- Thing description: Surfboard
- Number: 1

Made by Dr John

## Filtering records

To filter records you can add a WHERE clause (use AND/ OR if needed). An example is shown below:

```
28 $sql = "SELECT * FROM things WHERE description LIKE'Bicycle'";
29 }
30 $result = mysqli_query($con,$sql);
31 //output result
32 if(mysqli_num_rows($result)==0) //no records found
33 {
34     echo "<p>no records in DB".mysqli_num_rows($result)."</p>";
35 }
36 else // records were found in DB
37 {
```

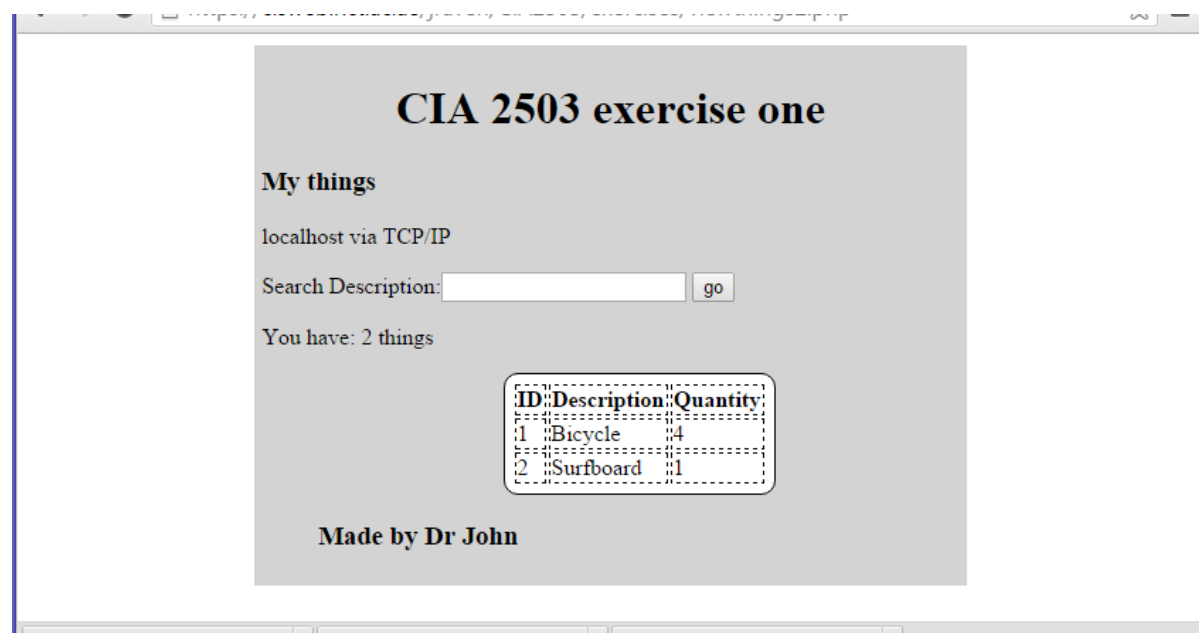
**Exercise :** Select only the Bicycle in the database as per the above. Then try any with quantity <5

**Challenge exercise:** Add a Search box

You can also add a form with a search box for pages like this on specific fields. It works the same as the filter command above but with the SELECT command changed to something like this:

```
60 else if(isset($_GET['searchby']))
61 {
62     $sql = "SELECT * FROM things WHERE description LIKE'%" . $_GET['searchby'] . "%'";
63 }
```

Of course you will also need the form, textbox and submit button added to the page (see below).



**CIA 2503 exercise one**

**My things**

localhost via TCP/IP

Search Description:

You have: 2 things

ID	Description	Quantity
1	Bicycle	4
2	Surfboard	1

**Made by Dr John**

### Notes:

- Set the form method to get and action to the same page.
- Use a submit button.

## Sorting records

Records can be sorted using an "ORDER BY" statement in the SELECT command followed by ASC or DESC. You can also add a link on your page that sends your sort command back to the page and then an if command that uses it if it is sent. Here is an example:

```
17 //connected
18 ?>
19 <a href="viewthings2.php?sortby=quantity">sort by quantity</a>
20 <?php
21 //run SQL query based on any sort string
22 if($_GET['sortby']=="quantity")
23 {
24 $sql = "SELECT * FROM things ORDER BY quantity ASC";
25 }
26 else
27 {
28 $sql = "SELECT * FROM things";
29 }
30 $result = mysqli_query($con,$sql);
31 //output result
```

See how the hyperlink sends the information back to the same page (view things) and if the URL query is sent then it will be picked up by the PHP and then the appropriate SELECT statement is used?

**Exercise:** Add a second hyperlink that sorts the things by description in descending order and then a third that sort the data by quantity.

**Challenge exercise:** now make the output look better by using a table with the sort hyperlinks in the top row of the table – something like the following (add styles also to improve the design).

### CIA 2503 MYSQL exercise

**My things**

localhost via TCP/IP

Search Description:

You have: 2 things

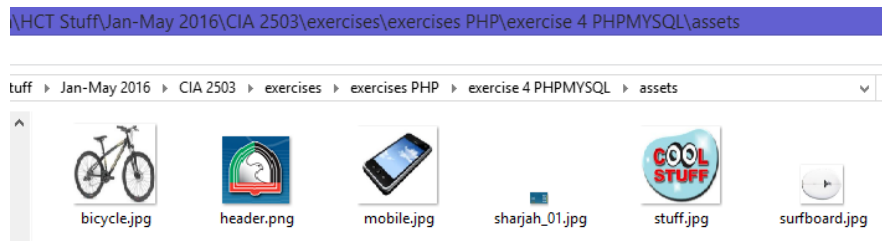
	Description	Quantity
ID	<a href="#">ascending</a>	<a href="#">ascending</a>
	<a href="#">descending</a>	<a href="#">descending</a>
1	Bicycle	4
2	Surfboard	1

Made by Dr John

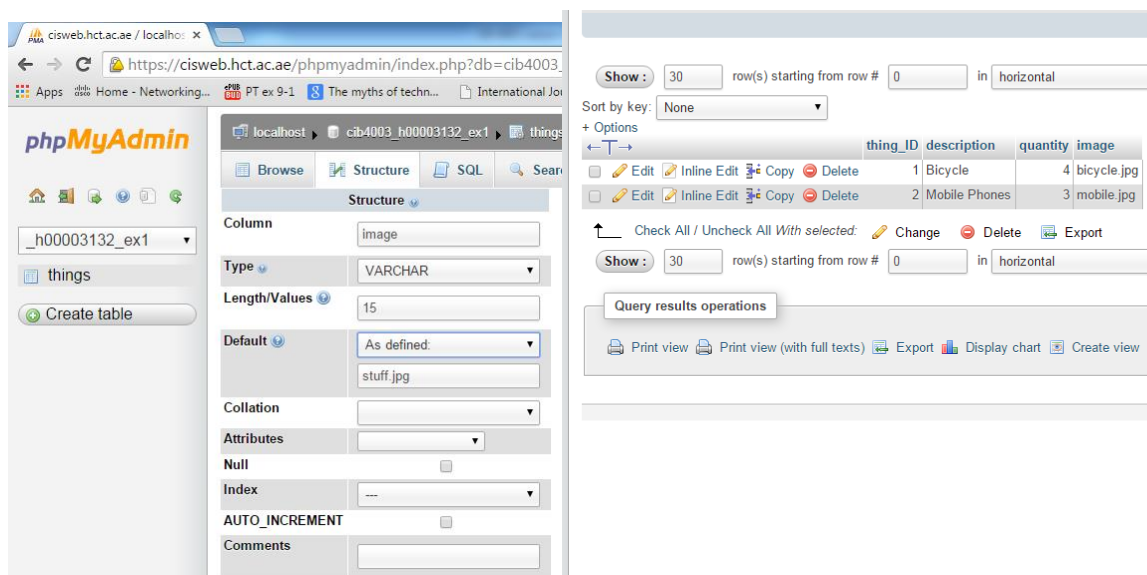
## Adding Images

Many want to have images of items stored in Database but the best way using MYSQL is to just store the name of the image and then have the images stored in an assets folder.

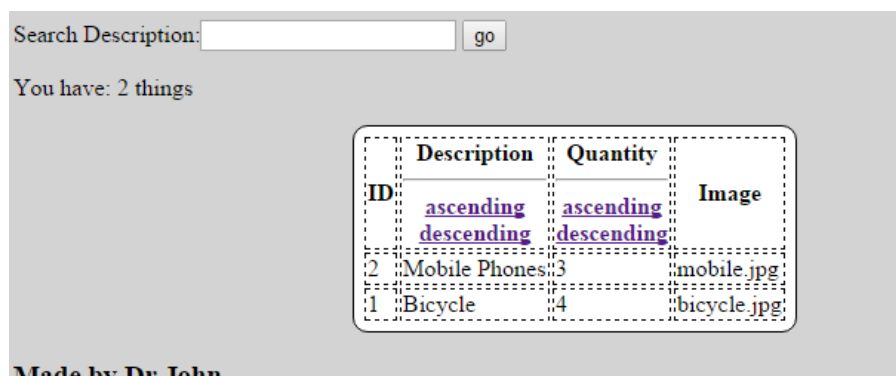
**First:** add your images to the assets folder in your web site and optimize them for the web and make sure they are consistent size. It's a good idea to have a default image.



Second add a column for the image name (varchar) and set the default image



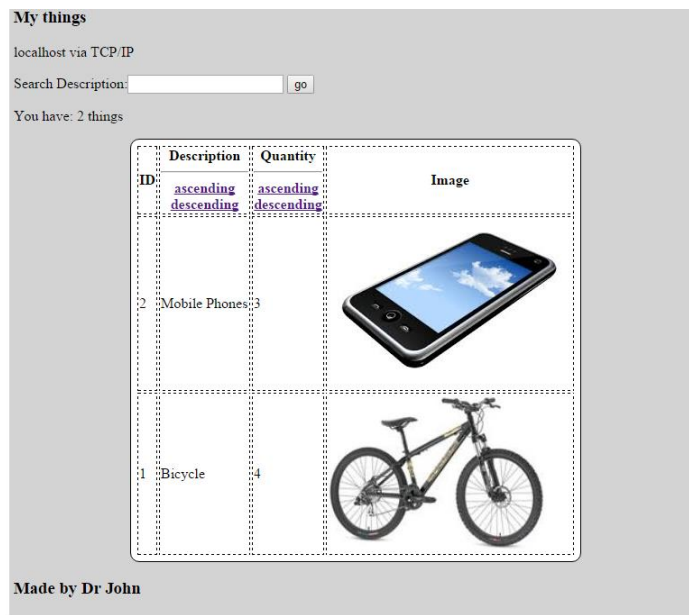
**Next:** in your viewthings.php display the image name in the web page from the DB – the same as the other values. Then upload the php and check it works:



**Finally:** copy the images to the assets folder on the server and alter the html code to include the image tag with the name of the image coming from the DB. Like this:

```
echo "<td>".$row['quantity']. "</td>";  
echo "<td><img src='assets/".$row['image']."' /></td>";  
echo "</tr>";  
} //end of loop
```

**Then:** upload and test to see the images.



**Note:** the problem with this method is that every image that is listed in the DB records must be in the assets folder on the server.

**Exercise:** Add images to your web page as per the above.

**Next topic:** Authentication and personalization