Online Experiments
Tutorial 02 – Send Stuff to the Server

Uri Hertz, PhD

Department of Cognitive Sciences

School of Psychological Science, University of Haifa

uhertz@cog.Haifa.ac.il

www.socialdecisionlab.net

אוניברסיטת חיפה
University of Haifa
جامعة حيفا

Social Decision Making Lab

Web based experiments

Client

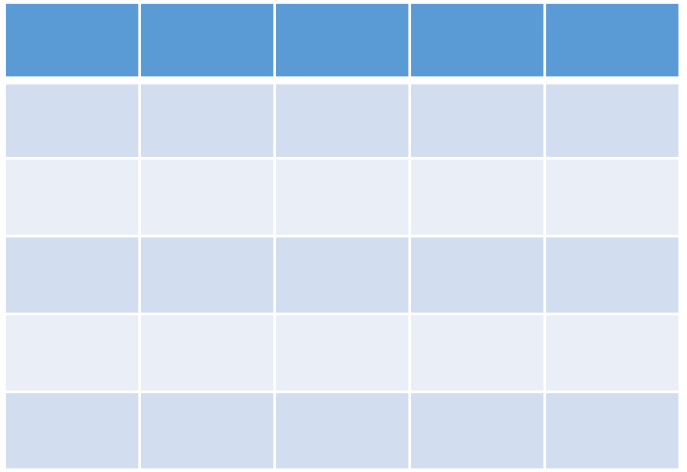
Using browser to compile web pages.
Including html, JS, CSS, images and other files.
Runs client side scripts.

Server

Stores the content.
Stores database.
Run server side scripts.

Web based experiments

Database



Files for client

Php
Node.js

SQL
MySQL
SQLlite
MongoDB
mariadb

Server and Database

I use php scripts as server side scripts to communicate with the database.

In some cases (multiplayer games) I used node.js – a javascript server side.

I use MySQL as a database. I did not explore other options yet. However, as I carry most of my data analysis using tables (in R), SQL tables are suitable.

How to store data

How do you plan to analyse the data? Which software? Which analyses?

This will determine the most suitable way to save the data.

You may also want to consider what data to store – things that are easy to calculate during the experiment, but is a pain to dig out post-hoc.

How to store data

I use R and dplyr, tibbles and data frames, where storing every trial in a separate row is useful, and the columns include different factors.

I also carry many within-subjects analyses, so it is important for me to track subjects across the experiment.

ID	TrialNo	Choice	Side	Reward	RT
1111	0	1	-1	1	1234
1111	1	1	1	0	2344
1111	2	2	1	1	2333
1111	3	2	-1	0	1243
1111	4	1	-1	1	2344

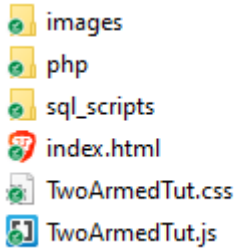
Source files

You can access the additional source files in:

Note that there are now two additional folders – sql and php.

The sql folder contains the sql scripts for creating the database tables.

The Php folder contains the php scripts that communicate with the database and the client.



SQL

"SQL", the abbreviation for Structured Query Language.

Manages relational tables, i.e. one may relate to others.

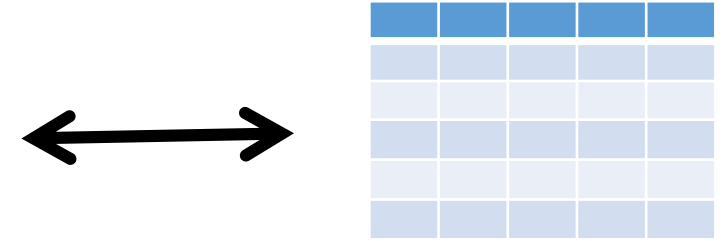
It is stored on a server, where you can use some management software to keep track of data, build and edit databases and tables.

Here is SQ L code to build the table for our experiment:

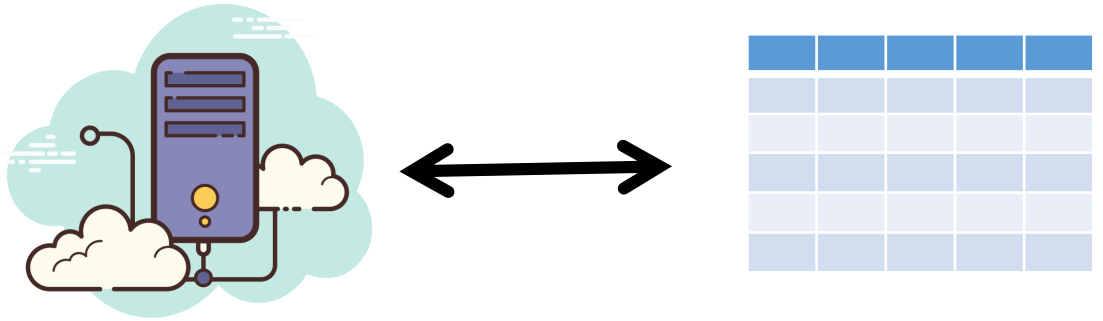
```
CREATE TABLE `survey_experiment` (
  `id` varchar(20) NOT NULL,
  `trialnum` int(11) DEFAULT NULL,
  `choice` int(11) DEFAULT NULL,
  `side` int(11) DEFAULT NULL,
`reward` int(11) DEFAULT NULL,
  `rt` int(11) DEFAULT NULL,
  `time` varchar(20) DEFAULT NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

Sending data to the database

- The server runs php compiler, and can execute our requests.
- We can get information from the client.
- We can run sql requests within the script.
- We can send information back.

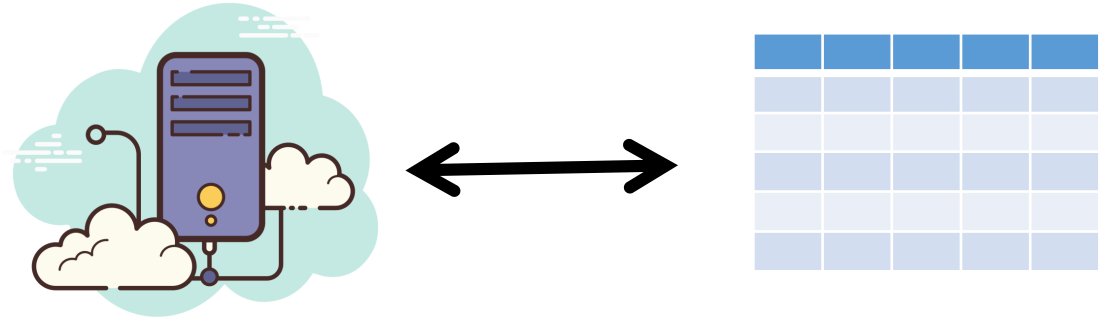


Sending data to the database InsertTrialData.php



```
<?php

include 'connectDB.php';

$ID = stripslashes(htmlspecialchars($_POST['ID']));
$TrialNum = stripslashes(htmlspecialchars($_POST['TrialNum']));
$Choice = stripslashes(htmlspecialchars($_POST['Choice']));
$Side = stripslashes(htmlspecialchars($_POST['Side']));
$Reward = stripslashes(htmlspecialchars($_POST['Reward']));
$RT = stripslashes(htmlspecialchars($_POST['RT']));
$Time = stripslashes(htmlspecialchars($_POST['Time']));

$stmt = $db->prepare("INSERT INTO two_arm_tut_data VALUE(?,?,?,?,?,?,?)");

$stmt->bind_param("siiiiii", $ID,$TrialNum,$Choice,$Side,$Reward,$RT,$Time);
$stmt->execute();
$err = $stmt->errno ;
$data[] = array(
     'ErrorNo' => $err,
   );
$stmt->close();
 $db->close();
echo json_encode($data);
 ?>
```

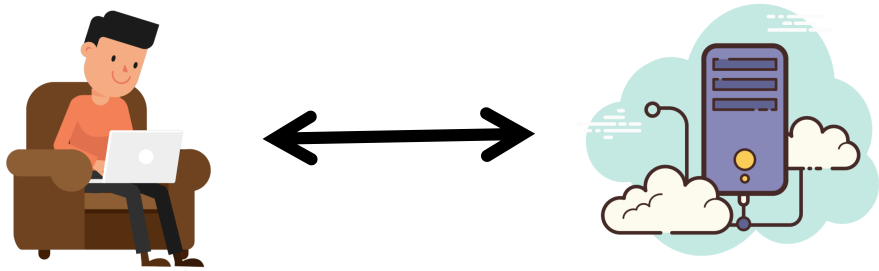
Sending data to the database connectDB.php



You can separate code that may be different from one deployment to another (for example your local server and live server), so you don't need to change all the code in each deployment.

```
<?php

$database="experiment1";
$host="localhost";
$user="wampuser";
$password="password";

$db = new mysqli($host, $user, $password, $database);

if (mysqli_connect_errno()) {
   printf("DB error: %s", mysqli_connect_error());
   exit();
}
```

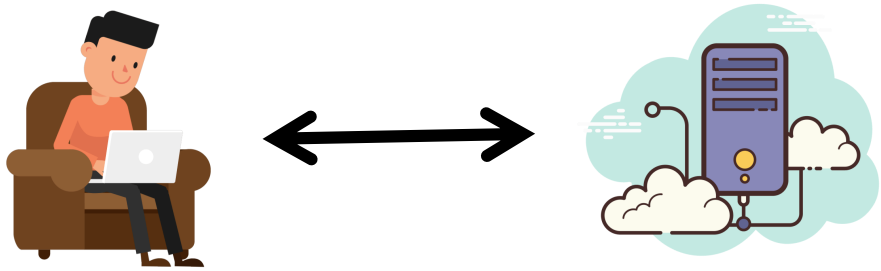
Sending data to php



AJAX stands for Asynchronous **JavaScript** And XML. In a nutshell, it is the use of the XMLHttp Request object to communicate with servers.

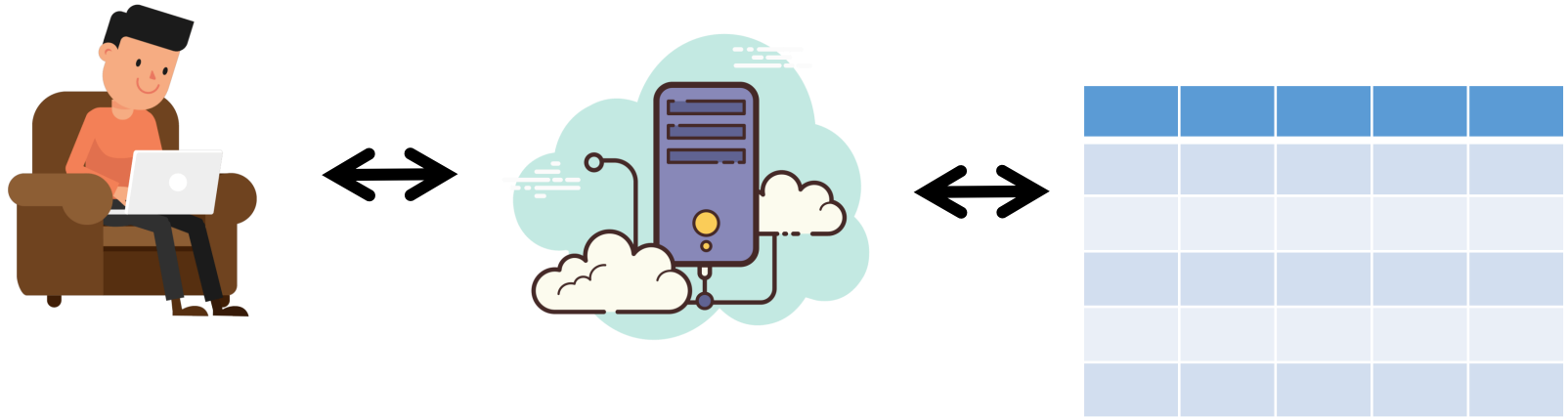
- Read data from a web server - after the page has loaded
- Update a web page without reloading the page
- Send data to a web server - in the background

Sending data to php – back to TwoArmedTut.js



```
function InsertDataAjax(TrialNum,Choice,Side,RT,Reward){
    var  ThisTime = (new Date()).getTime();

    $.ajax({
        type: 'POST',
        data: {ID:SubID,TrialNum:TrialNum,Choice:Choice,Side:Side,Reward:Reward,RT:RT,Time:ThisTime},
        async: false,
        url: 'php/InsertTrialData.php',
        dataType: 'json',
        success: function(r) {
            if (r[0].ErrorNo > 0) {
                Error();
            }
        }, error: function(XMLHttpRequest, textStatus, errorThrown) {
            alert("Status: " + textStatus);
            alert("Error: " + errorThrown);
        }
     });
}
```

Set up your local development environment



We know how all these connections work – but how to actually set it up?

We need to build a local server – our computer can be a server and a client at the same time!

We want it to be able to run php, hold a database, and communicate with the browser (with the ajax).

Set up your local development environment

I use WAMP (for mac users – XAMP)
https://sourceforge.net/projects/wampserver/

You can also try MAMP or any other solution.

It creates a server on your computer, with MySQL database.

Once activated, you can see the local server in the browser by going to

http://localhost/

You can upload files to the server's folder, usually c:/wamp64/www

Simply copy and paste a folder and access it through the localhost.

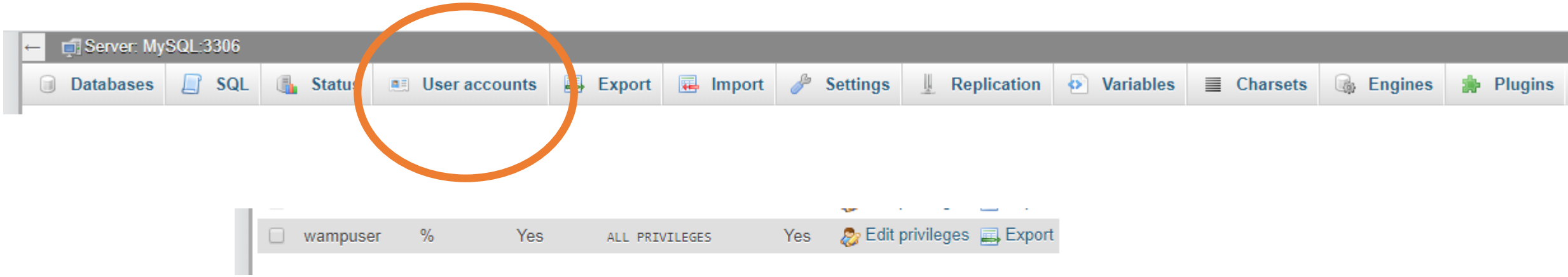
Set up your local development environment

You can access you database by going to:

http://localhost/phpmyadmin/

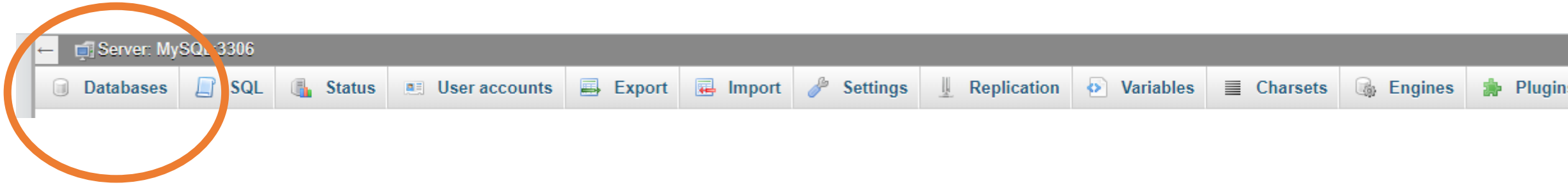
And initially log in using the user: root and no password.

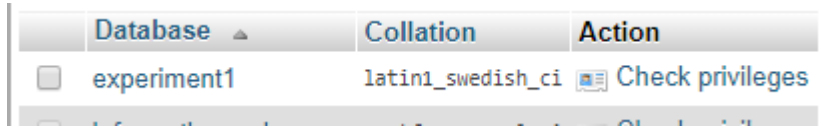
Once inside, create a user (I called mine wampuser) and a password (mine is password)



Set up your local development environment

Next, you'll need to add a database – database is a collection of many tables. Mine is called experiment1 for reasons.



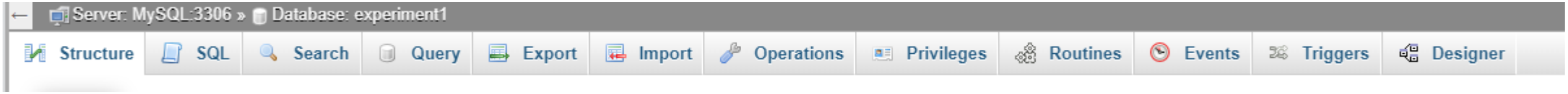


Set up your local development environment

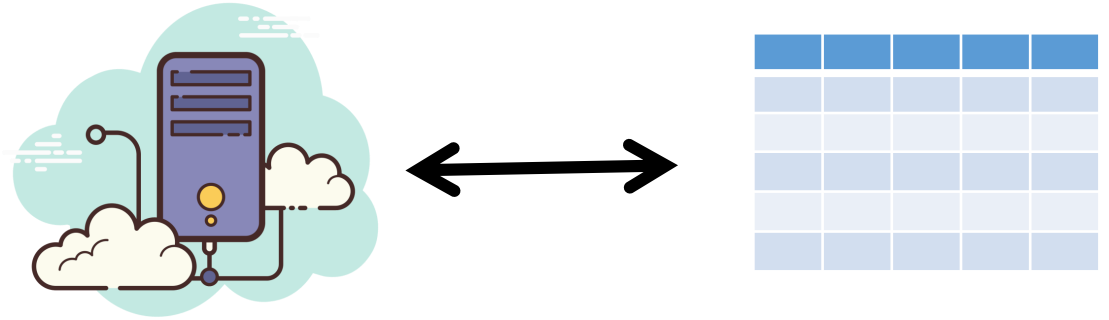
Now, choose the database you created from the left side menu, and create your tables.

You can do it in ‘Structure’ – add new

Or run the sql script in the SQL section.



Sending data to the database connectDB.php



Make sure the parameters in the connectDB.php file are in line with your settings.

```
<?php

$database="experiment1";
$host="localhost";
$user="wampuser";
$password=“password";

$db = new mysqli($host, $user, $password, $database);

if (mysqli_connect_errno()) {
   printf("DB error: %s", mysqli_connect_error());
   exit();
}
```

Running and debuging

You can tell your IDE to run your experiment on a the local server.

In Brackets you set it up in ‘project settings’, directing it to run on http://localserver/yourexperimentfoldername

You can also use ‘inspect’ in the browser to see how information is moving to the server, and to catch errors in the php scripts, by clicking on the php script in the network tab.

What else?

You can add more ajaxes and tables, to store more data.

You can also use php to export data more easily.

I included another ajax in the js file that sends the participant's ID to a 'finished' table once the participant finishes the experiment.

Another php file can be open in the browser to present all the participants that finished the experiment, with a link to download their data.

Pressing the link triggers another php file that generates a csv file with the data.

Finished AJAX

```
function InsertFinishedAjax(){

  $.ajax({
          type: 'POST',
          data: {ID:SubID},
          async: false,
          url: 'php/FinishCode.php',
          dataType: 'json',
          success: function(r) {
             if (r[0].ErrorNo > 0) {
                Error();
             }
          }, error: function(XMLHttpRequest, textStatus, errorThrown) {
             alert("Status: " + textStatus);
             alert("Error: " + errorThrown);
          }
       });

}
```

FinishCode.php

```
<?php

include 'connectDB.php';
$ID = stripslashes(htmlspecialchars($_POST['SubID']));

$stmt = $db->prepare("INSERT INTO two_arm_tut_finished VALUE(?,NOW())");
$stmt->bind_param("s", $ID );
$stmt->execute();
$err = $stmt->errno ;
$data[] = array(
      'ErrorNo' => $err,
    );
$stmt->close();
 $db->close();
echo json_encode($data);
 ?>
```

Present_Subjects.php

```
<?php

include 'connectDB.php';

$query = "select * from two_arm_tut_finished;";
 if ($result = mysqli_query($db, $query)){

 Print "<table border cellpadding=3>";

  while ($row = mysqli_fetch_array($result))  {
   echo "<th>ID:</th> <td>".$row['ID'] . "</td> ";
 echo "<th>Time:</th> <td>".$row['time'] . " </td>";

 echo " <td><a href=Export_Table.php?SubCode=".$row['ID']
.">Export Data</a></td>". "</tr>";

                }
                echo "</table>";
 }

  mysqli_free_result($result);
          mysqli_close($db);
?>
```

This is a different type of php file:

We now getting data from a table, and not the other way around.

We present the data in an html table.

We run this php file directly in the browser, and not using ajax to call it.

Export_Table.php

```
<?php
include 'connectDB.php';

$Subject = stripslashes(htmlspecialchars($_GET['SubCode']));

$query = "SELECT  * FROM two_arm_tut_data WHERE ID= '". $Subject ."' ;";

 if ($result = mysqli_query($db, $query)){

$pasajeros = "id,trialnum,choice,side,reward,rt,time"."\r\n";//these are the columns names

  while ($row = mysqli_fetch_array($result))  {
       $pasajeros .= $row["id"] ."," .$row["trialnum"] ."," .$row["choice"] ."," .$row["side"] ."," .$row["reward"] ."," .$row["rt"] ."," .$row["time"]."\r\n"; //note
the comma here
                  }
 }
$filename = "pasajeros_" . date("Y-m-d_H-i");
header("Content-type: application/vnd.ms-excel");
header("Content-disposition: csv" . date("Y-m-d") . ".csv");
header("Content-disposition: filename=TwoArmTut_" . $Subject . ".csv");
 echo $pasajeros;
  mysqli_free_result($result);
   mysqli_close($db);
?>
```

Enhancements

You can use the same mechanism to collect additional data and store it in different tables:

Demographics

Questionnaires

Amazon worker ID (to check for duplications)

Export an entire table (the export_table with no selection by ID)

Create a live interaction?