COMP353 Project #1
Warm-up Project

Mair Elbaz, 40004558
Daniel Vigny-Pau, 40034769
Francois David, 40046319
Alexandre Therrien, 40057134
Charles-Antoine Guite, 40063098

Professor: Dr. Desai

Fall 2019
Concordia University

Creating the Tables

Below are the lines of code that we have put in our program to create the various tables required for this project.

```
//query the database with the command in (). Creates table 'people'
  $mysqli->query("CREATE TABLE people (id smallint unsigned not null auto_increment, lastname varchar(20) not null, firstname varchar(20) not null, middle_name varchar(20), userID int not null, password int not null, constraint pk_people primary key (id) );");
```

This code queries our database to create a new table with the name ‘people’. That is, assuming you are already connected to your database and that you have selected a database to use. The $mysqli variable is the variable used to store the connection to our database. The characters in blue compose the query.

We are to create a table that has a variable id, which is never to be attributed the value null (from the constraint “not null”), that is always positive (“unsigned”), and that increments automatically (“auto_increment”) so that there are never two same id values. It is also a small int value, which means it has a smaller range than an int value. The last statement of this query (constraint pk_people primary key (id)) indicates that this variable will be used as the primary key.

The rest of the statements between commas are giving the different columns this new table will have. The varchar type is a type of data element that contains a series of characters. We have the ‘lastname’ element, which has the constraint to be not null and a maximum length of 20. We have the ‘firstname’ element, which has the constraint to be not null and a maximum length of 20. We have the ‘middle_name’ element, which has a maximum length of 20 and can be null.

Then we have two int values, userID and password. These are integer types, both of which have the constraint to be not null.

```
//query the database with the command in (). Creates table 'event'
  $mysqli->query("CREATE TABLE event (id smallint unsigned not null auto_increment, Event varchar(20), EventID int, start_date varchar(20), end_date varchar(20), AdminUserID int, constraint pk_event primary key (id));");
```

Similar to the previous table, this statement creates a table called ‘event’ and has the same primary key statements as in the previous example.

It creates the different columns Event (varchar of maximum length 20 that can be null), EventID (integer value that can be null), start_date (varchar of maximum length 20 that can be null), end_date (varchar of maximum length 20 that can be null), and AdminUserID (integer value that can be null).

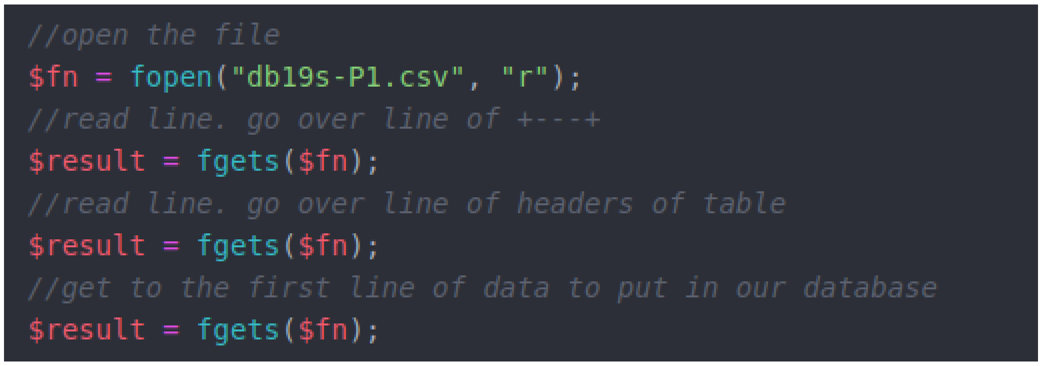
```
//query the database with the command in (). Creates table 'role_of_people_in_the_event'
  $mysqli->query("CREATE TABLE role_of_people_in_the_event (id smallint unsigned not null
auto_increment, userid int, EventID int, constraint pk_role primary key (id));");
```

Similar to the previous table, this statement creates a table called ‘role_of_people_in_the_event’ and has the same primary key statements as in the previous example.

This table has a userid column (integer, could be null) and an EventID column (integer, not null).

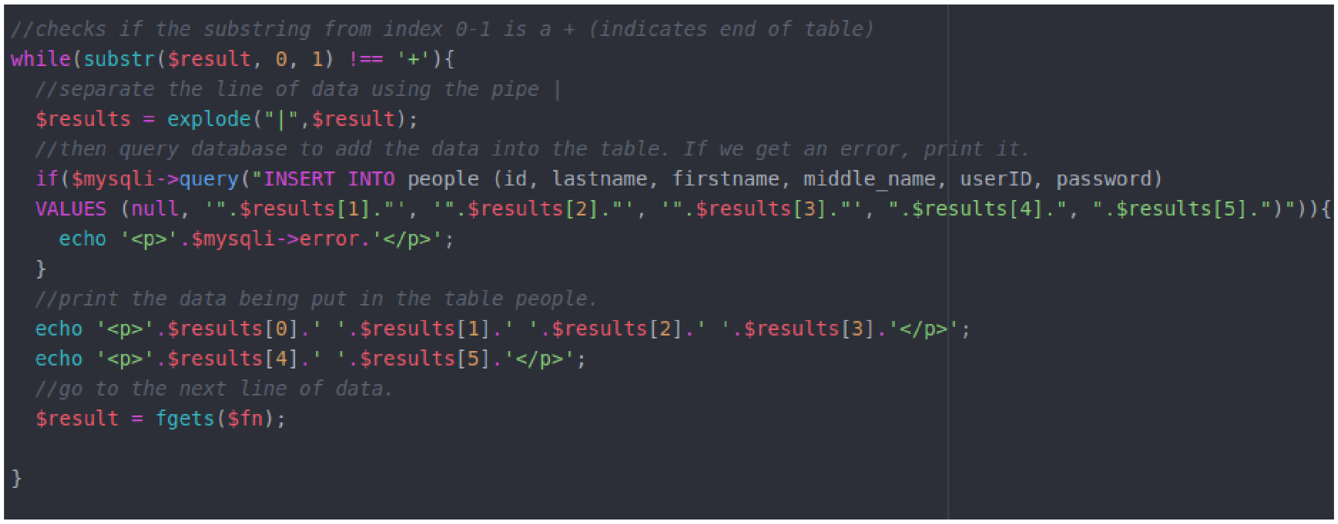
Prepare and Read Input Data

To start reading the input data, some operations had to be done before.



```php
//open the file
$fn = fopen("db19s-P1.csv", "r");
//read line. go over line of +---+
$result = fgets($fn);
//read line. go over line of headers of table
$result = fgets($fn);
//get to the first line of data to put in our database
$result = fgets($fn);
```

First, I needed to open the file in php with read authorization. Then, I placed my $result variable to the first line of data of the first table.

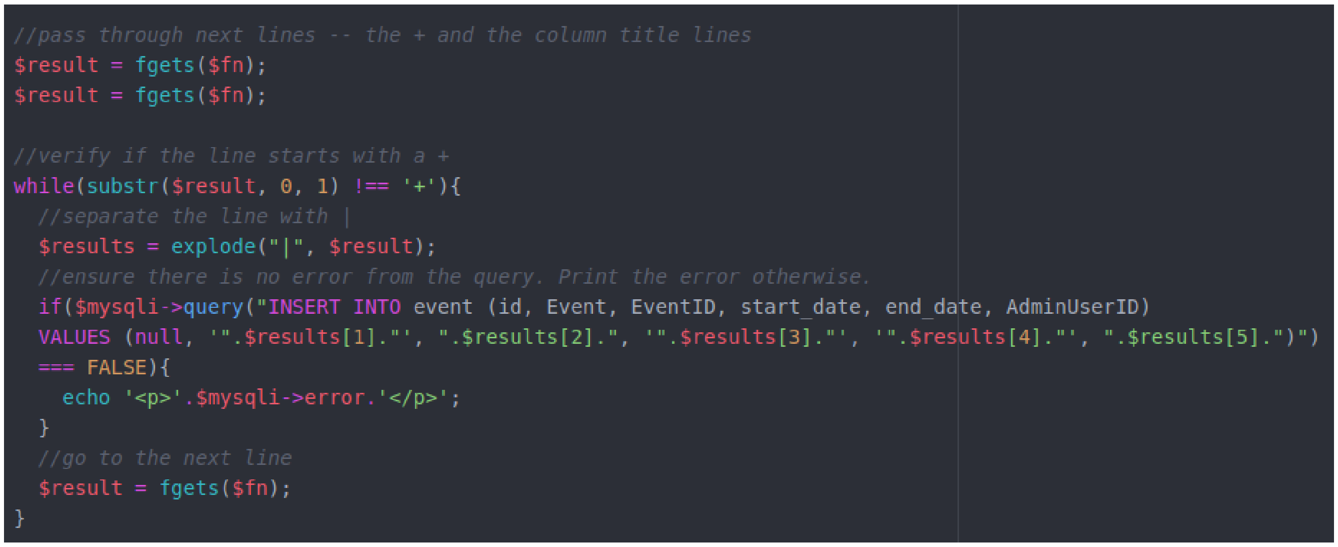


```php
//checks if the substring from index 0-1 is a + (indicates end of table)
while(substr($result, 0, 1) !== '+'){
  //separate the line of data using the pipe |
  $results = explode("|",$result);
  //then query database to add the data into the table. If we get an error, print it.
  if($mysqli->query("INSERT INTO people (id, lastname, firstname, middle_name, userID, password)
  VALUES (null, '".$results[1]."', '".$results[2]."', '".$results[3]."', ".$results[4].", ".$results[5].")")){
    echo '<p>'.$mysqli->error.'</p>';
  }
  //print the data being put in the table people.
  echo '<p>'.$results[0].' '.$results[1].' '.$results[2].' '.$results[3].'</p>';
  echo '<p>'.$results[4].' '.$results[5].'</p>';
  //go to the next line of data.
  $result = fgets($fn);

}
```

Then, I am doing a loop until I read a line starting with the + value (which is the line that separates the different tables from one another). To read all the different data to go in the different columns of the table, I separate the line of data received by using the pipe (|) character, which is what separates those values. Because the line starts with a pipe character, the first array element we are interested in is the index 1, as index 0 will always be an empty string.

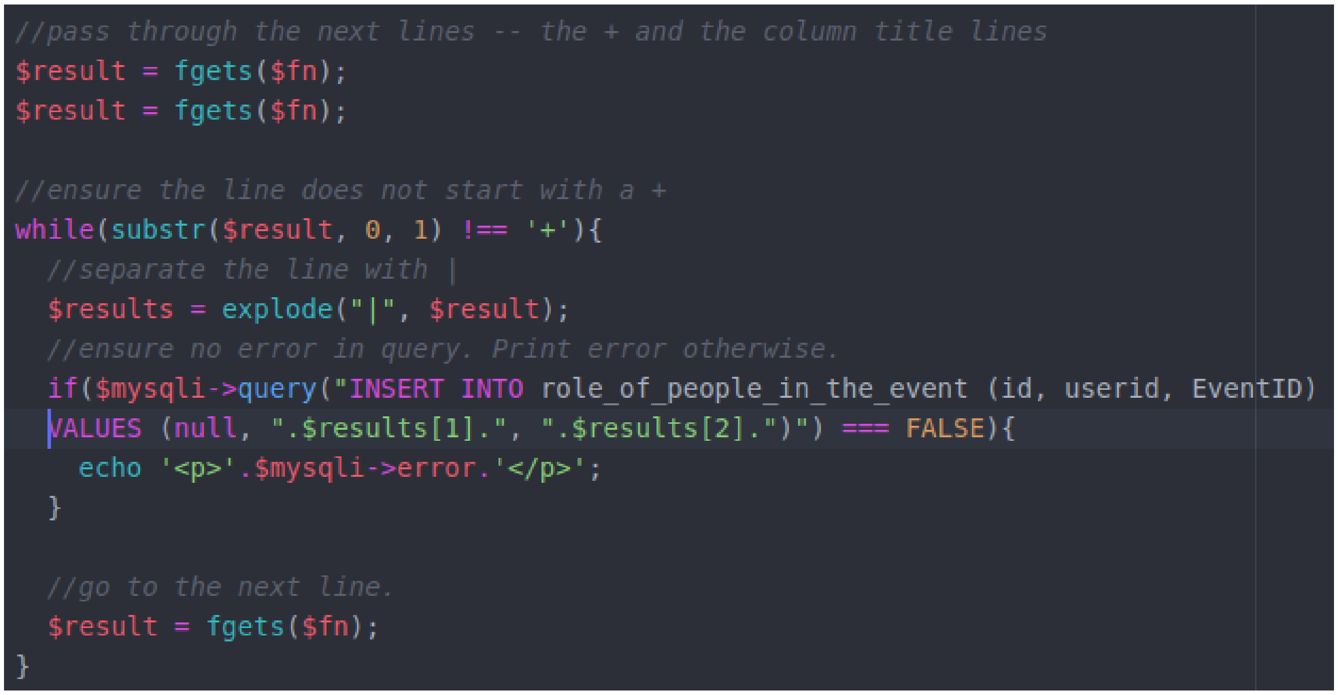
I follow that with an if condition that does the query and verifies that the output is not FALSE, which would indicate that there is an error in the query. If there is an error, I would print the error in the browser. This query indicates that we want to add some data to the table ‘people’ in the following order: (id, lastname, firstname, middle_name, userID, password). The INSERT INTO query adds that data into the table ‘people’. Then, we use VALUES (…) with the values in the same order as they were mentioned before.

Then the results of my separation using the pipe character are printed to the screen, and I follow that by going into the next line of the file doing $result = fgets($fn);


```
//pass through next lines -- the + and the column title lines
$result = fgets($fn);
$result = fgets($fn);

//verify if the line starts with a +
while(substr($result, 0, 1) !== '+'){
  //separate the line with |
  $results = explode("|", $result);
  //ensure there is no error from the query. Print the error otherwise.
  if($mysqli->query("INSERT INTO event (id, Event, EventID, start_date, end_date, AdminUserID)
  VALUES (null, '".$results[1]."', ".$results[2].", '".$results[3]."', '".$results[4]."', ".$results[5].")")
  === FALSE){
    echo '<p>'.$mysqli->error.'</p>';
  }
  //go to the next line
  $result = fgets($fn);
}
```

Next, once we have reached a line starting with a + (which separates the tables), we exit the loop from the previous image. We then go over that + line and the column titles' line using fgets, and we enter a new loop which is very similar to the previous one. The only difference is that the results are not getting printed and the query has been adapted to add the data into the 'event' table. This time I put the first result from the split as Event, then EventID, start_date, end_date, and finally AdminUserID.

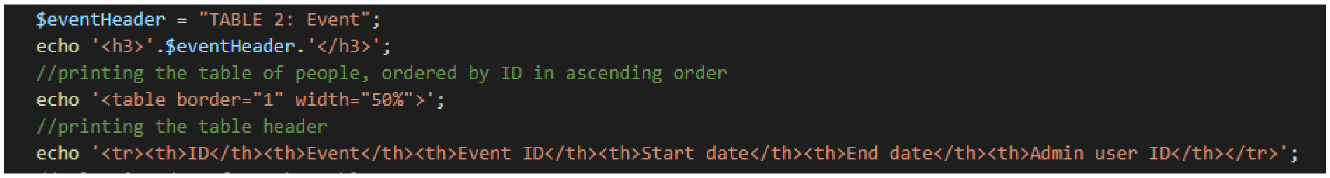
I then go to the next line.


```
//pass through the next lines -- the + and the column title lines
$result = fgets($fn);
$result = fgets($fn);

//ensure the line does not start with a +
while(substr($result, 0, 1) !== '+'){
  //separate the line with |
  $results = explode("|", $result);
  //ensure no error in query. Print error otherwise.
  if($mysqli->query("INSERT INTO role_of_people_in_the_event (id, userid, EventID)
  VALUES (null, ".$results[1].", ".$results[2].")") === FALSE){
    echo '<p>'.$mysqli->error.'</p>';
  }

  //go to the next line.
  $result = fgets($fn);
}
```

Same process here. Once we reach the line starting with + separating the tables, we exit the loop in the previous image and enter this one after going over the + and the column titles' lines. This time I add the content of the split into the 'role_of_people_in_the_event'.

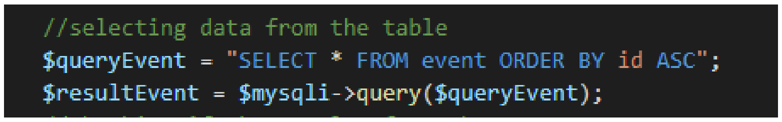
Accessing the data

We begin by first creating our table and its header, creating a column for each corresponding field in the tables we have created earlier. Take for example the table `event`.



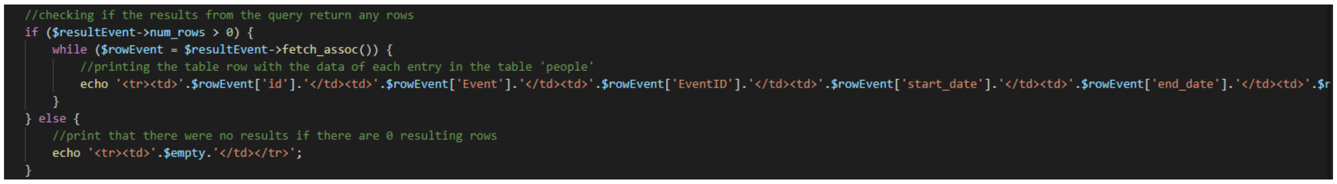
```
$eventHeader = "TABLE 2: Event";
echo '<h3>'.$eventHeader.'</h3>';
//printing the table of people, ordered by ID in ascending order
echo '<table border="1" width="50%">';
//printing the table header
echo '<tr><th>ID</th><th>Event</th><th>Event ID</th><th>Start date</th><th>End date</th><th>Admin user ID</th></tr>';
```

We define a title and print it, followed by our opening tags for our table, along with its table header.



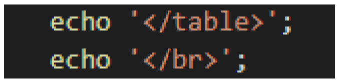
```
//selecting data from the table
$queryEvent = "SELECT * FROM event ORDER BY id ASC";
$resultEvent = $mysqli->query($queryEvent);
```

To access the data, we create an SQL query where we select all the data from a table, such as the table `event` we have previously created. `*` denotes that we are selecting all the data, whilst `ORDER BY id ASC` indicates that we are ordering the entries in the table in ascending order by ID.



```
//checking if the results from the query return any rows
if ($resultEvent->num_rows > 0) {
    while ($rowEvent = $resultEvent->fetch_assoc()) {
        //printing the table row with the data of each entry in the table 'people'
        echo '<tr><td>'.$rowEvent['id'].'</td><td>'.$rowEvent['Event'].'</td><td>'.$rowEvent['EventID'].'</td><td>'.$rowEvent['start_date'].'</td><td>'.$rowEvent['end_date'].'</td><td>'.$r
    }
} else {
    //print that there were no results if there are 0 resulting rows
    echo '<tr><td>'.$empty.'</td></tr>';
}
```

Now we have two cases: if the number of rows returned by our previous call is more than 0, we will display the results in the table. If there are no results from our query, we will print that there are no results. Using `fetch_assoc()`, we can iterate through each entry of our query results and print a table row for each one.



```
echo '</table>';
echo '</br>';
```

Once we have finished printing our table rows, we can close the tags of our table. This gives us the following table when we execute the code above:

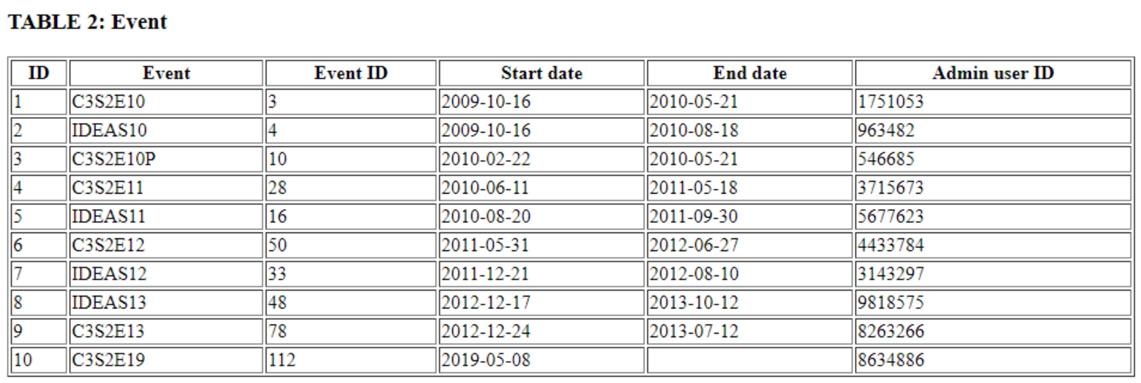
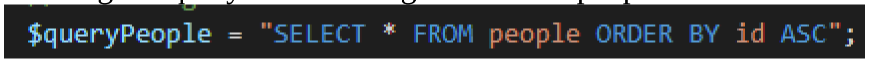


TABLE 2: Event

ID	Event	Event ID	Start date	End date	Admin user ID
1	C3S2E10	3	2009-10-16	2010-05-21	1751053
2	IDEAS10	4	2009-10-16	2010-08-18	963482
3	C3S2E10P	10	2010-02-22	2010-05-21	546685
4	C3S2E11	28	2010-06-11	2011-05-18	3715673
5	IDEAS11	16	2010-08-20	2011-09-30	5677623
6	C3S2E12	50	2011-05-31	2012-06-27	4433784
7	IDEAS12	33	2011-12-21	2012-08-10	3143297
8	IDEAS13	48	2012-12-17	2013-10-12	9818575
9	C3S2E13	78	2012-12-24	2013-07-12	8263266
10	C3S2E19	112	2019-05-08		8634886

However, by modifying our query, we can obtain a variety of different results.

Our original query for obtaining the table of people was:



```
$queryPeople = "SELECT * FROM people ORDER BY id ASC";
```

This one selects all entries from the table `people` and orders them by `id` in ascending order.

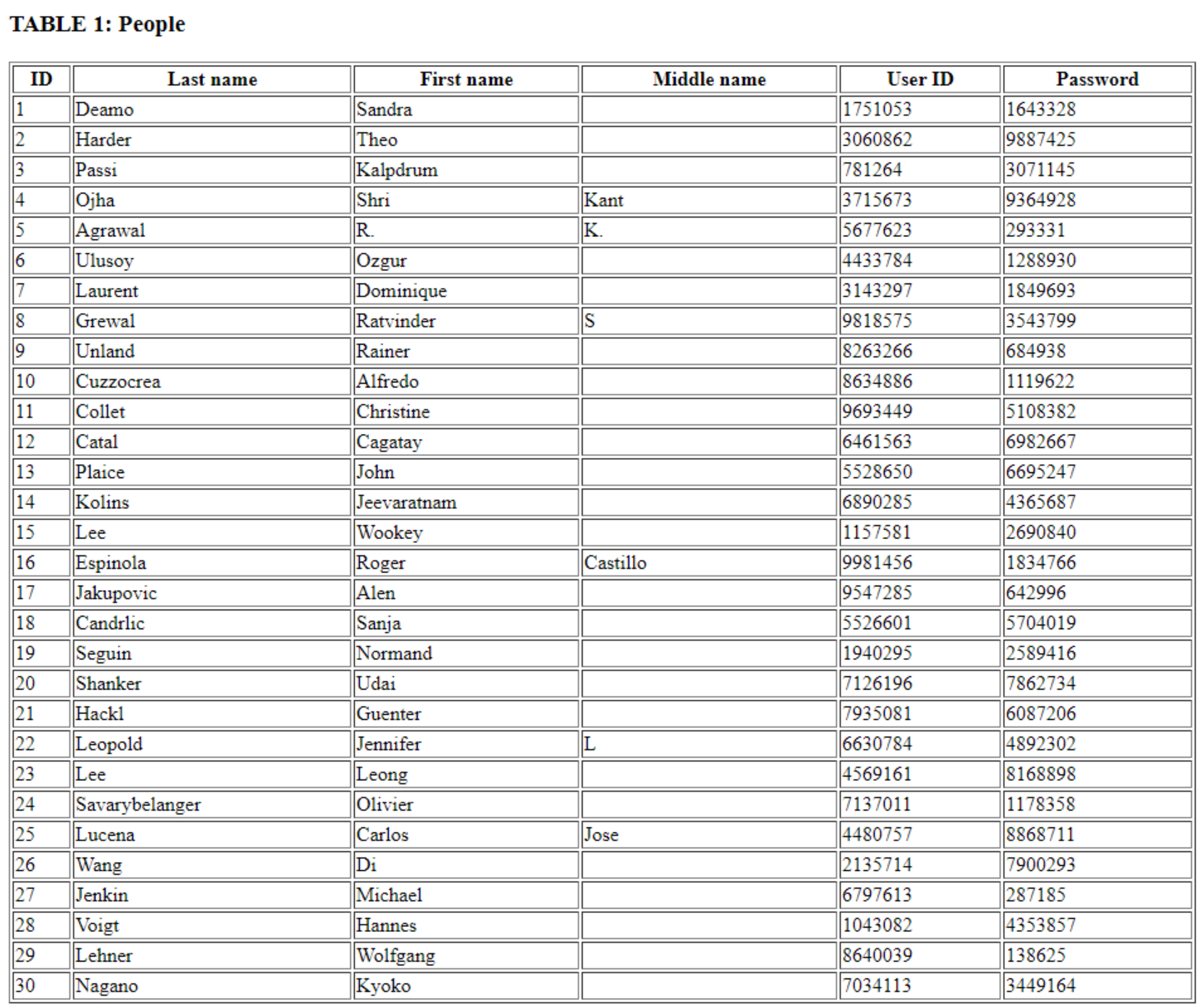
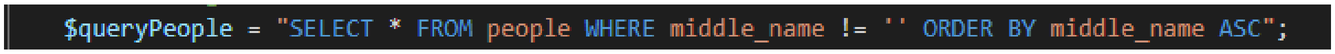


TABLE 1: People

ID	Last name	First name	Middle name	User ID	Password
1	Deamo	Sandra		1751053	1643328
2	Harder	Theo		3060862	9887425
3	Passi	Kalpdrum		781264	3071145
4	Ojha	Shri	Kant	3715673	9364928
5	Agrawal	R.	K.	5677623	293331
6	Ulusoy	Ozgur		4433784	1288930
7	Laurent	Dominique		3143297	1849693
8	Grewal	Ratvinder	S	9818575	3543799
9	Unland	Rainer		8263266	684938
10	Cuzzocrea	Alfredo		8634886	1119622
11	Collet	Christine		9693449	5108382
12	Catal	Cagatay		6461563	6982667
13	Plaice	John		5528650	6695247
14	Kolins	Jeevaratnam		6890285	4365687
15	Lee	Wookey		1157581	2690840
16	Espinola	Roger	Castillo	9981456	1834766
17	Jakupovic	Alen		9547285	642996
18	Candrlic	Sanja		5526601	5704019
19	Seguin	Normand		1940295	2589416
20	Shanker	Udai		7126196	7862734
21	Hackl	Guenter		7935081	6087206
22	Leopold	Jennifer	L	6630784	4892302
23	Lee	Leong		4569161	8168898
24	Savarybelanger	Olivier		7137011	1178358
25	Lucena	Carlos	Jose	4480757	8868711
26	Wang	Di		2135714	7900293
27	Jenkin	Michael		6797613	287185
28	Voigt	Hannes		1043082	4353857
29	Lehner	Wolfgang		8640039	138625
30	Nagano	Kyoko		7034113	3449164

However we can change this query to a number of different ones, such as the following:



```
$queryPeople = "SELECT * FROM people WHERE middle_name != '' ORDER BY middle_name ASC";
```

Which would return only people with a middle name, or rather where their field for `middle_name` is not empty, sorted by their middle name in alphabetical order.

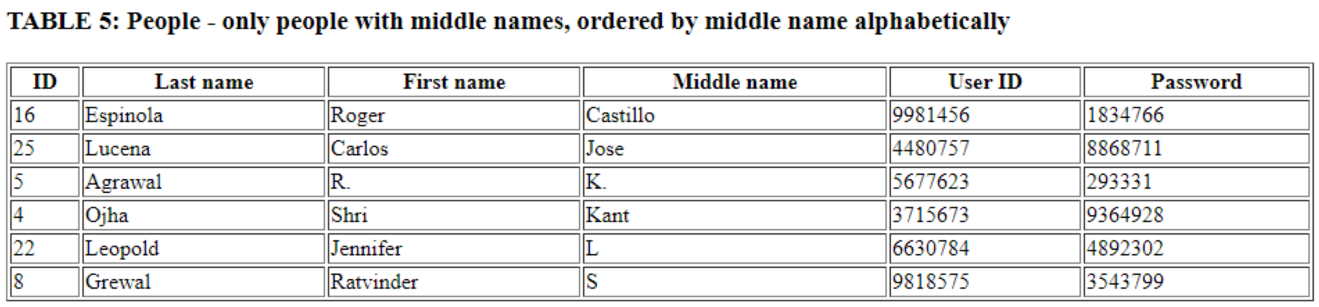
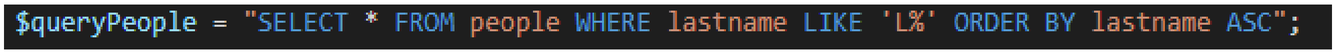


TABLE 5: People - only people with middle names, ordered by middle name alphabetically

ID	Last name	First name	Middle name	User ID	Password
16	Espinola	Roger	Castillo	9981456	1834766
25	Lucena	Carlos	Jose	4480757	8868711
5	Agrawal	R.	K.	5677623	293331
4	Ojha	Shri	Kant	3715673	9364928
22	Leopold	Jennifer	L	6630784	4892302
8	Grewal	Ratvinder	S	9818575	3543799

Or consequently, if the query were changed to something like:



```
$queryPeople = "SELECT * FROM people WHERE lastname LIKE 'L%' ORDER BY lastname ASC";
```

This query selects all entries from the table `people` with a `lastname` field beginning with the letter `L`, essentially returning all people with a last name starting with L, sorted by last name in alphabetical order. The `%` symbol means that the character `L` can be followed by anything.

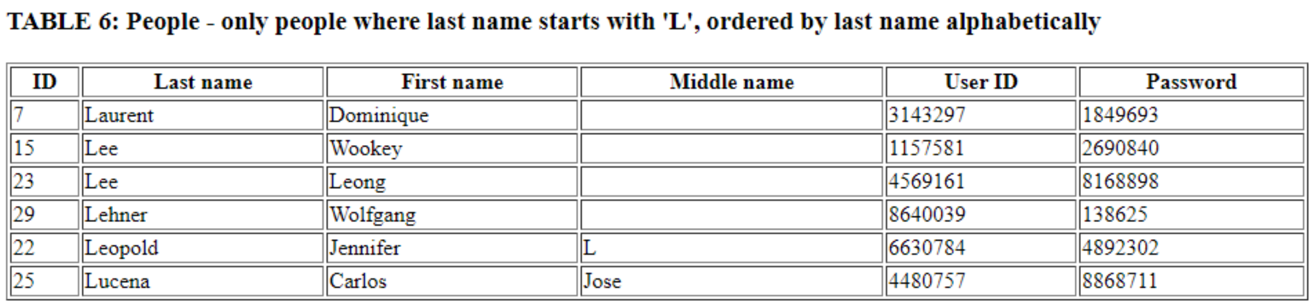
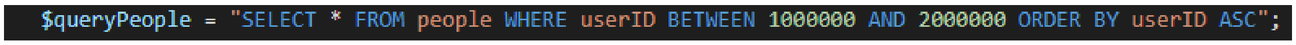


TABLE 6: People - only people where last name starts with 'L', ordered by last name alphabetically

ID	Last name	First name	Middle name	User ID	Password
7	Laurent	Dominique		3143297	1849693
15	Lee	Wookey		1157581	2690840
23	Lee	Leong		4569161	8168898
29	Lehner	Wolfgang		8640039	138625
22	Leopold	Jennifer	L	6630784	4892302
25	Lucena	Carlos	Jose	4480757	8868711

This third query below will select all entries from people with a user ID between 1000000 and 2000000, ordered by user ID in ascending order.



```
$queryPeople = "SELECT * FROM people WHERE userID BETWEEN 1000000 AND 2000000 ORDER BY userID ASC";
```

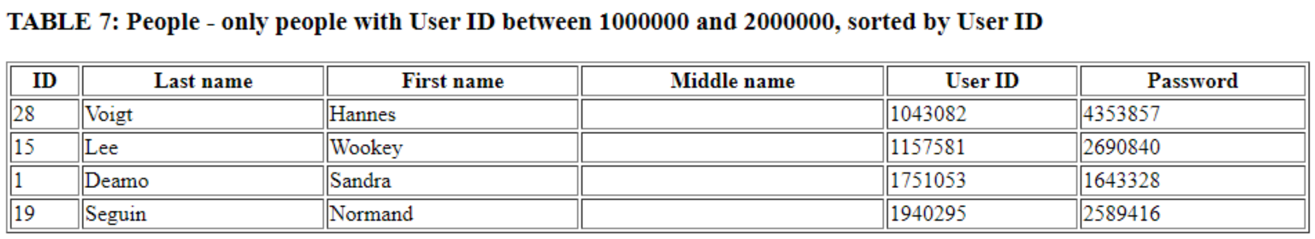
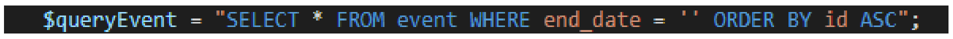


TABLE 7: People - only people with User ID between 1000000 and 2000000, sorted by User ID

ID	Last name	First name	Middle name	User ID	Password
28	Voigt	Hannes		1043082	4353857
15	Lee	Wookey		1157581	2690840
1	Deamo	Sandra		1751053	1643328
19	Seguin	Normand		1940295	2589416



```
$queryEvent = "SELECT * FROM event WHERE end_date = '' ORDER BY id ASC";
```

This query allows us to view all events with no end date, noted by entries where the `end_date` field is empty, sorted by ID in ascending order.

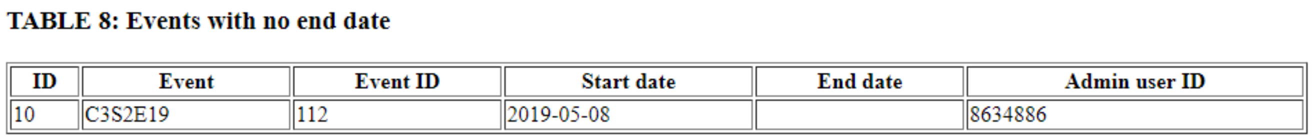
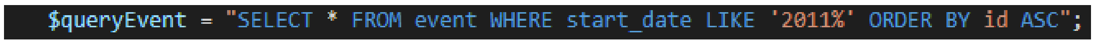


TABLE 8: Events with no end date

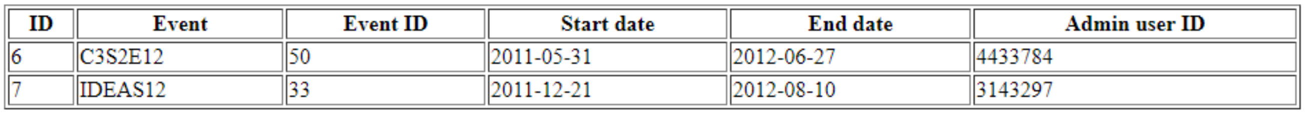
ID	Event	Event ID	Start date	End date	Admin user ID
10	C3S2E19	112	2019-05-08		8634886



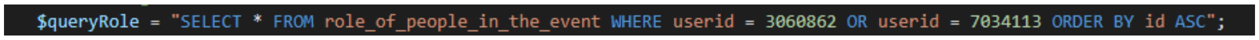
```
$queryEvent = "SELECT * FROM event WHERE start_date LIKE '2011%' ORDER BY id ASC";
```

This query allows us to view all events where the start date is in 2011, noted by entries where the `start_date` field begins with 2011 and is followed by anything else pertaining to the date, thanks to the `%` wildcard character, as mentioned above.

TABLE 9: Events starting in the year 2011



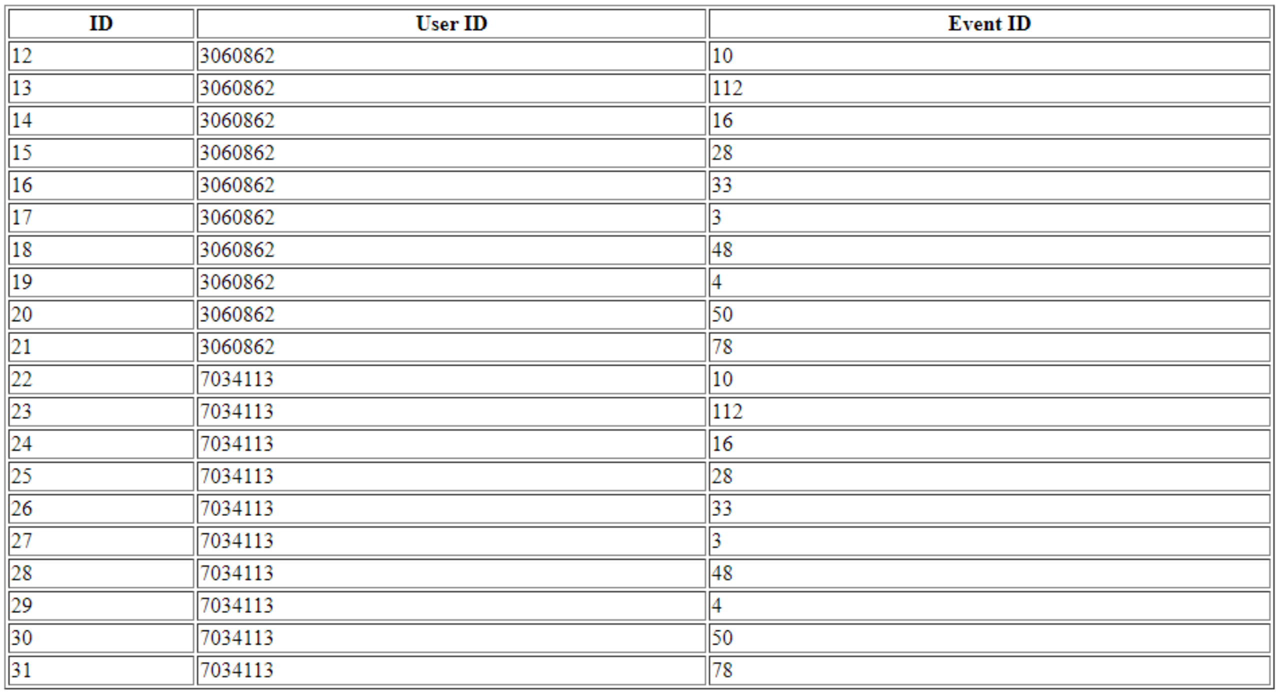
ID	Event	Event ID	Start date	End date	Admin user ID
6	C3S2E12	50	2011-05-31	2012-06-27	4433784
7	IDEAS12	33	2011-12-21	2012-08-10	3143297



```
$queryRole = "SELECT * FROM role_of_people_in_the_event WHERE userid = 3060862 OR userid = 7034113 ORDER BY id ASC";
```

This query allows us to select all entries in the `role_of_people_in_the_event` table with user IDs 3060862 or 7034113, thanks to the `OR` key word.

TABLE 10: Role of people in the event only with users 3060862 and 7034113



ID	User ID	Event ID
12	3060862	10
13	3060862	112
14	3060862	16
15	3060862	28
16	3060862	33
17	3060862	3
18	3060862	48
19	3060862	4
20	3060862	50
21	3060862	78
22	7034113	10
23	7034113	112
24	7034113	16
25	7034113	28
26	7034113	33
27	7034113	3
28	7034113	48
29	7034113	4
30	7034113	50
31	7034113	78

```
    $queryRole = "SELECT * FROM role_of_people_in_the_event WHERE EventID = 4 ORDER BY id ASC";
```

This last query selects all entries in `role_of_people_in_the_event` where the `EventID` field is equal to 4.

TABLE 11: Role of people in the event only with Event ID 4

ID	User ID	Event ID
9	2135714	4
19	3060862	4
29	7034113	4
39	781264	4
48	9547285	4

We can even modify the table itself and select only specific field names. If our header is only the following columns:

```
   echo '<tr><th>Last name</th><th>First name</th><th>Middle name</th></tr>';
```

We can consequently select only the corresponding ones, being `firstname`, `lastname`, and `middle_name`, sorting them by `lastname` in descending order:

```
   $queryPeople = "SELECT firstname, lastname, middle_name FROM people ORDER BY lastname DESC";
```

And finally printing the corresponding smaller number of rows:

```
   echo '<tr><td>'.$rowPeople['lastname'].'</td><td>'.$rowPeople['firstname'].'</td><td>'.$rowPeople['middle_name'].'</td></tr>';
```

Giving us a final table of:

TABLE 4: People - only displaying first name, last name, and middle name, sorted by last name reverse alphabetically

Last name	First name	Middle name
Wang	Di	
Voigt	Hannes	
Unland	Rainer	
Ulusoy	Ozgur	
Shanker	Udai	
Seguin	Normand	
Savarybelanger	Olivier	
Plaice	John	
Passi	Kalpdrum	
Ojha	Shri	Kant
Nagano	Kyoko	
Lucena	Carlos	Jose
Leopold	Jennifer	L
Lehner	Wolfgang	
Lee	Leong	
Lee	Wookey	
Laurent	Dominique	
Kolins	Jeevaratnam	
Jenkin	Michael	
Jakupovic	Alen	
Harder	Theo	
Hackl	Guenter	
Grewal	Ratvinder	S
Espinola	Roger	Castillo
Deamo	Sandra	
Cuzzocrea	Alfredo	
Collet	Christine	
Catal	Cagatay	
Candrlic	Sanja	
Agrawal	R.	K.