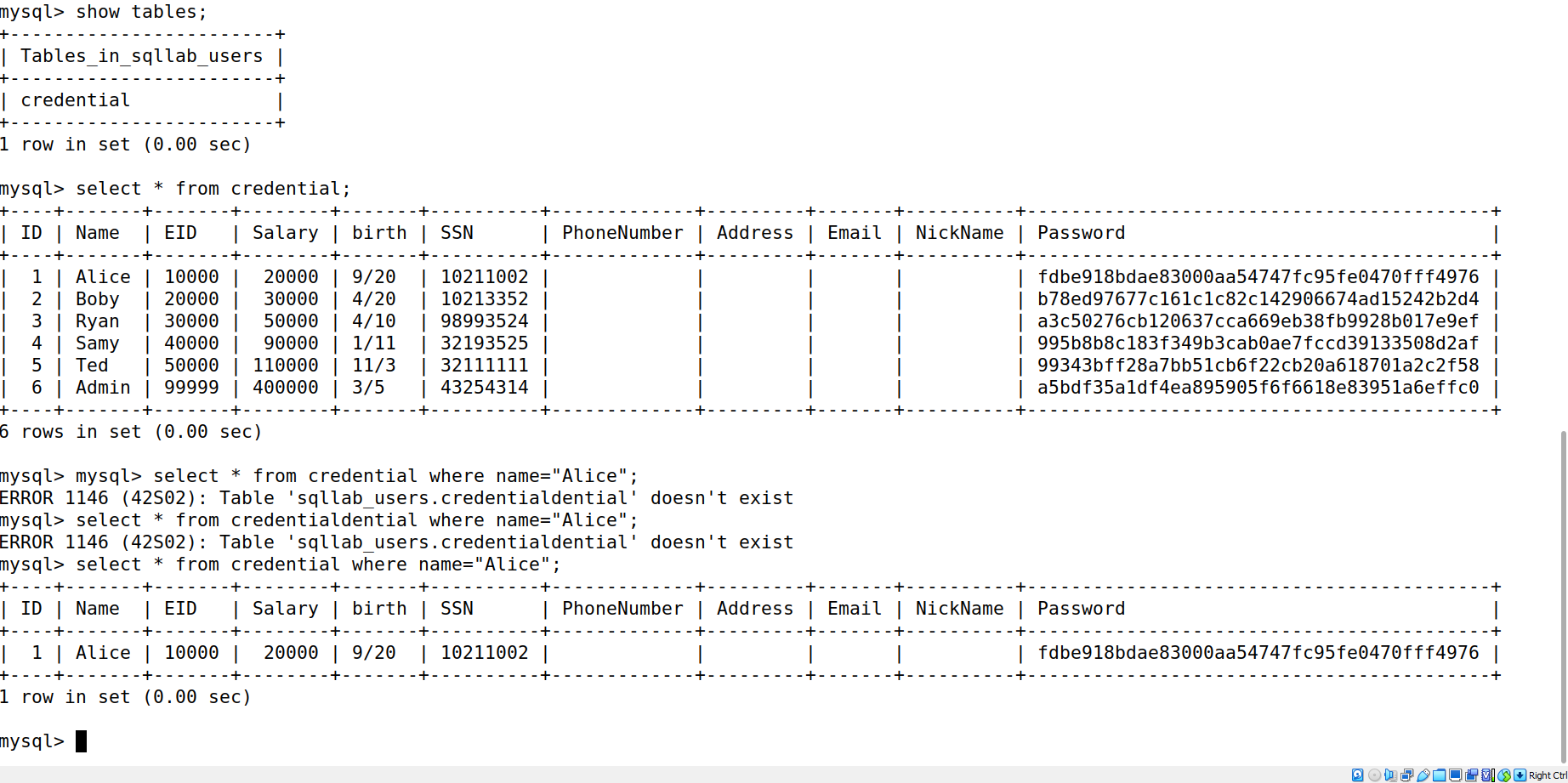
Name: Donghao Li

SUID: dli106

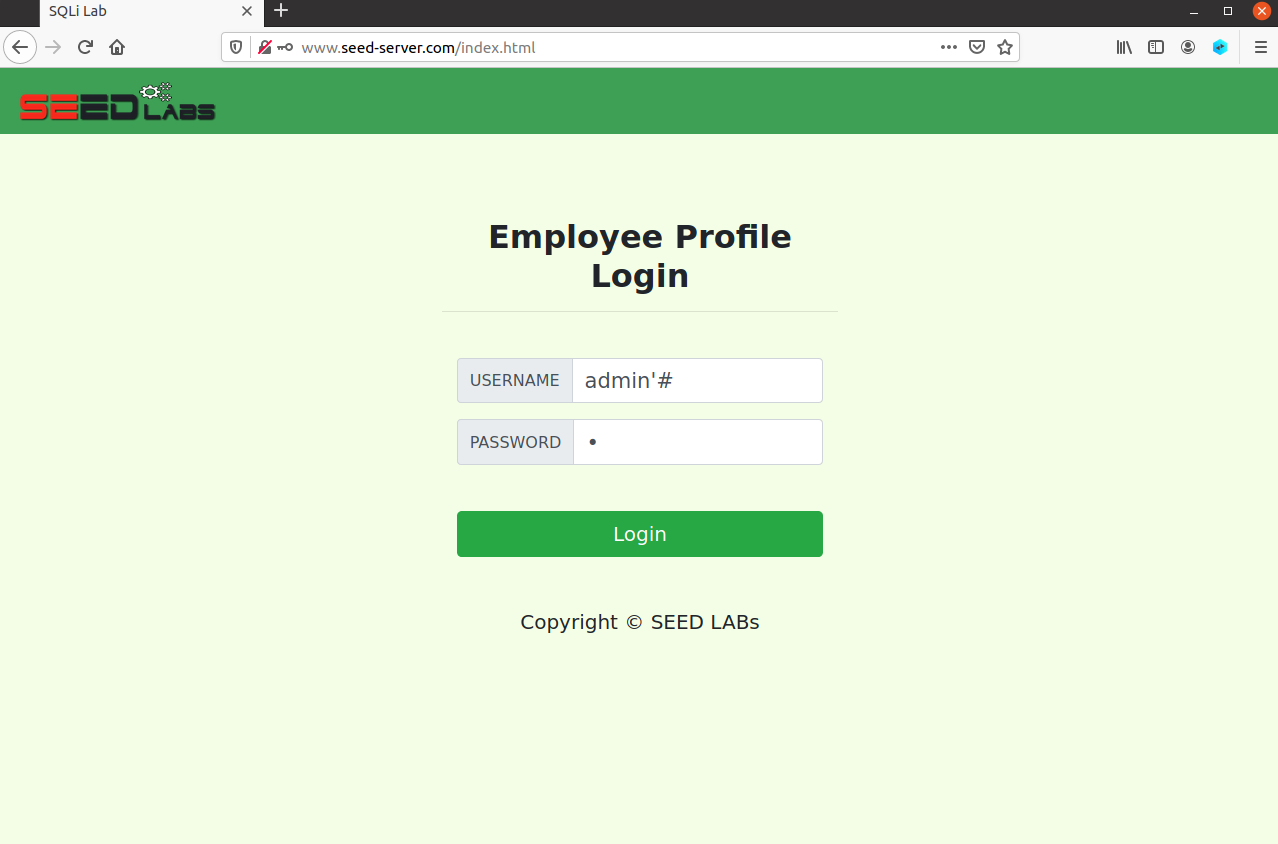
Task 1: Get Familiar with SQL Statements

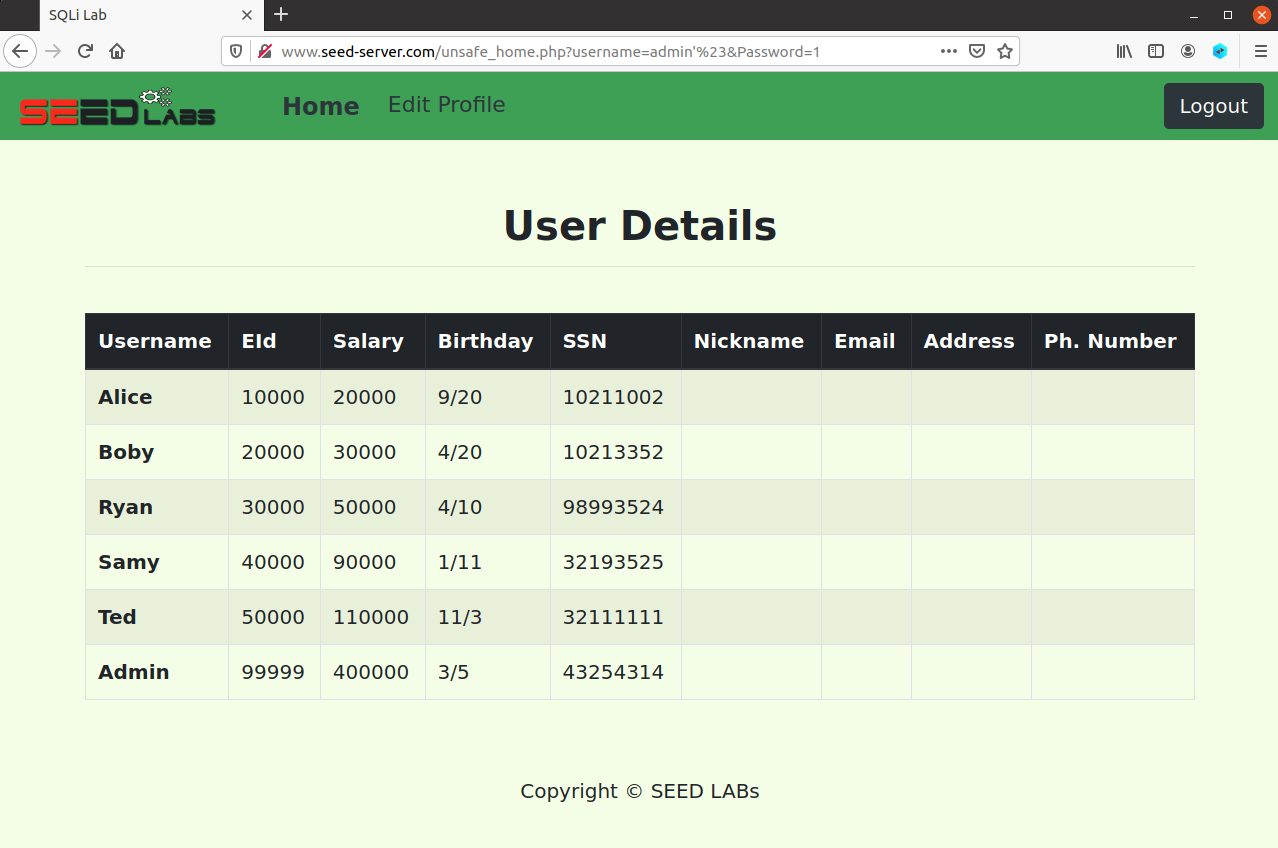


**As shown above, here is the table about Alice inside the credential database, each line represents Alice and her following information.**

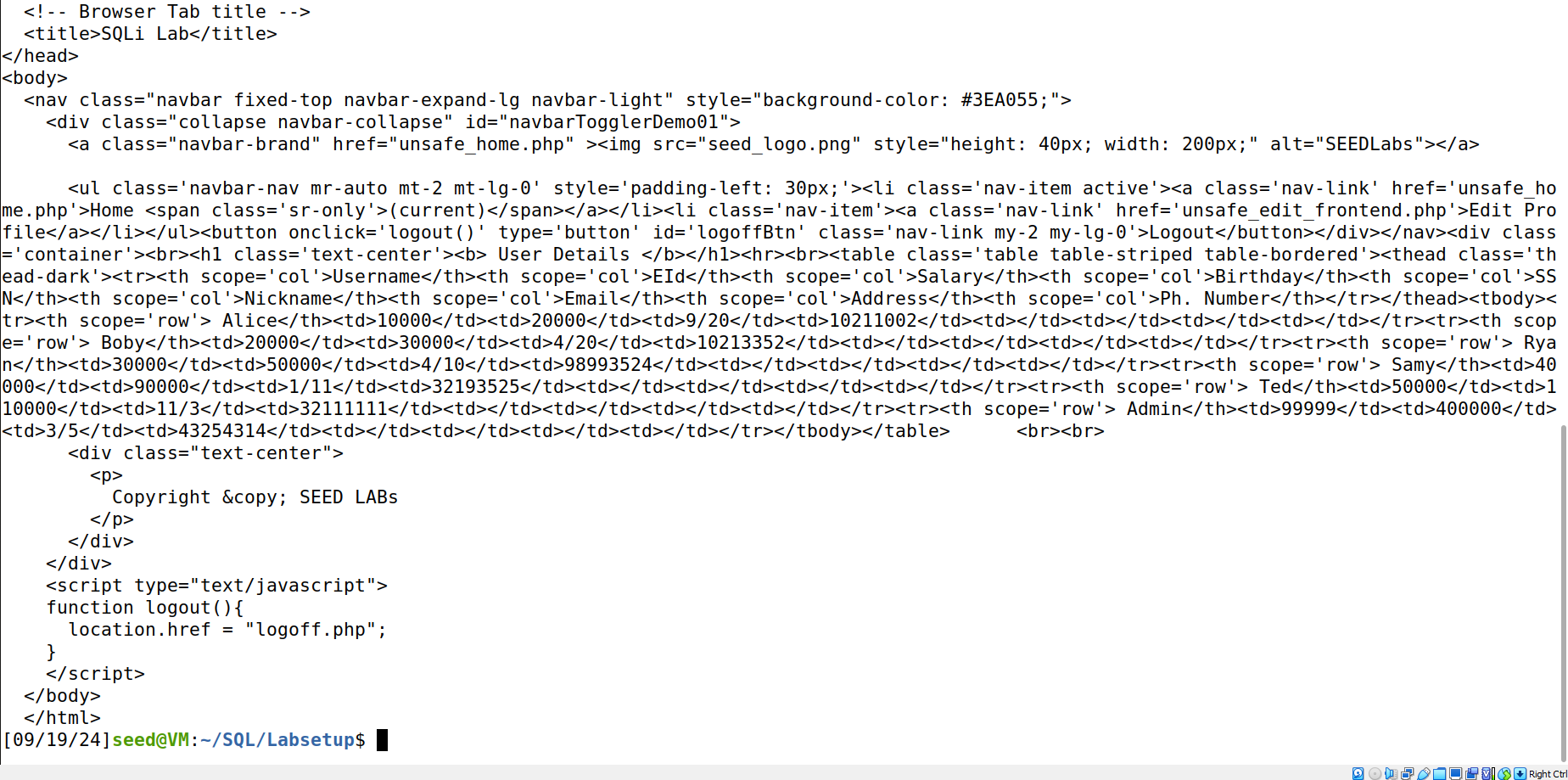
Task 2: SQL Injection Attack on SELECT Statement

Task 2.1: SQL Injection Attack from webpage.



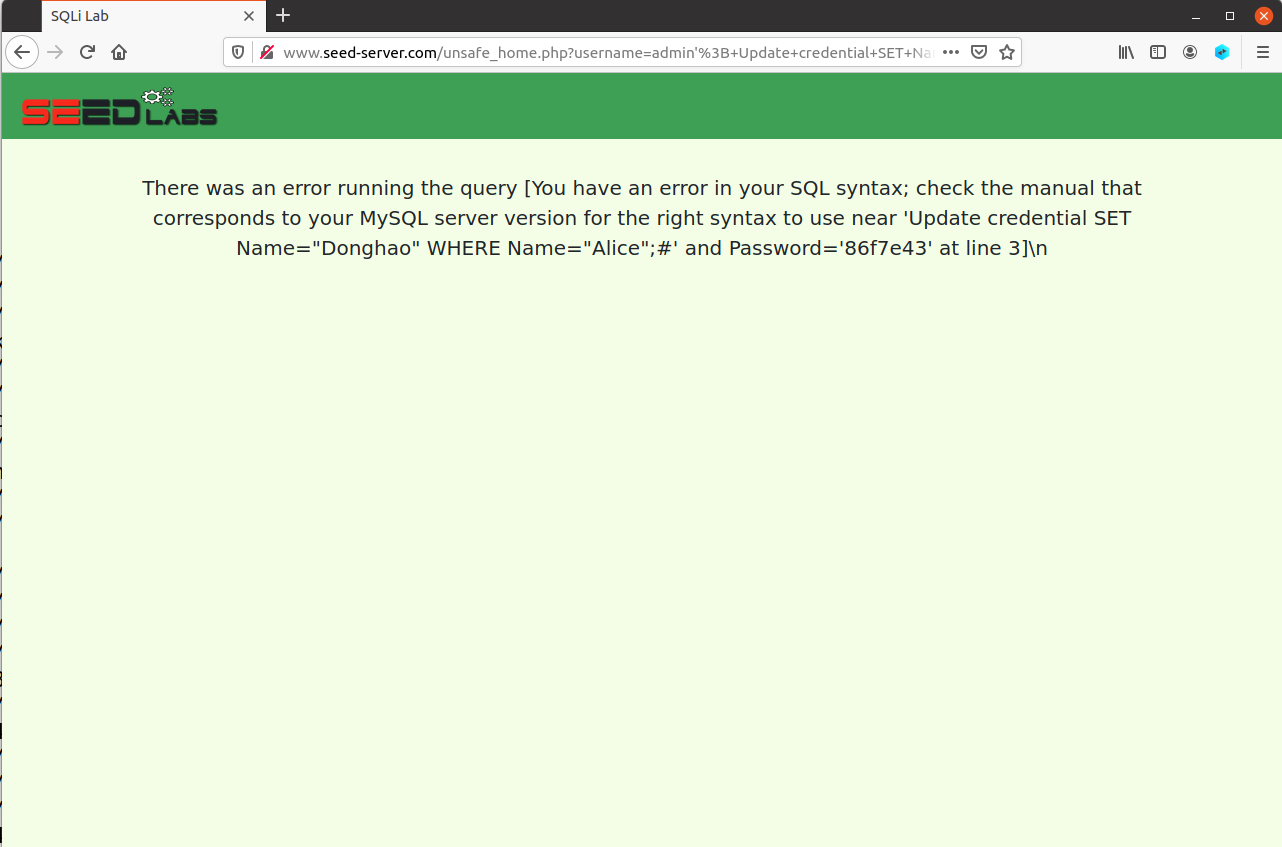
  
**Answer: As shown in figure one, when the attacker wants to log into the account of the administer without the password, the attacker would type ’# in the user name part what it do is that it will comment the rest of the php command “WHERE name= ’$input\_uname’ and Password=’$hashed\_pwd’"; so that the attacker will be able to log in without the password.**

Task 2.2: SQL Injection Attack from command line.

  
curl ‘[www.seed-server.com/unsafe\_home.php?username=admin%27%23&Password=a](http://www.seed-server.com/unsafe_home.php?username=admin%27%23&Password=a)’

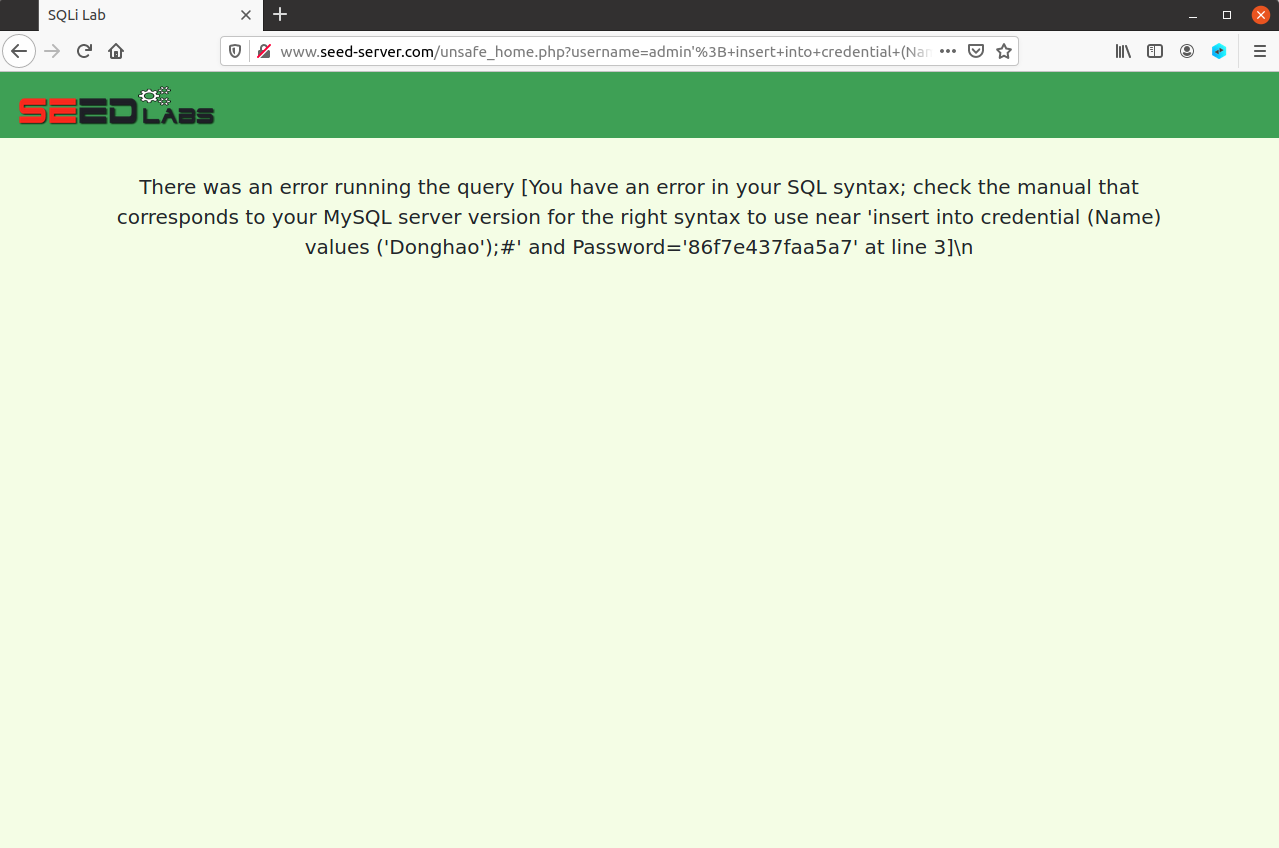
From the command view, we can also log into the admin account by comment out the password loop check. We can accomplish it by translating the “’#” into machine language which is %27%23(2\*16+7=39 which is the ‘ symbol, and 2\*16+3=35 which is # symbol), by this we can got a return in the form of html which should be read by the browser and translate to a webpage and show to the user. But we still can read necessary information from it.

Task 2.3: Append a new SQL statement



**Answer: admin'; Update credential SET Name="Donghao" WHERE Name="Alice";#  
I use the semicolon to separate the two commands but it is not work. Then I tried to insert and log in command, but it still not work.**

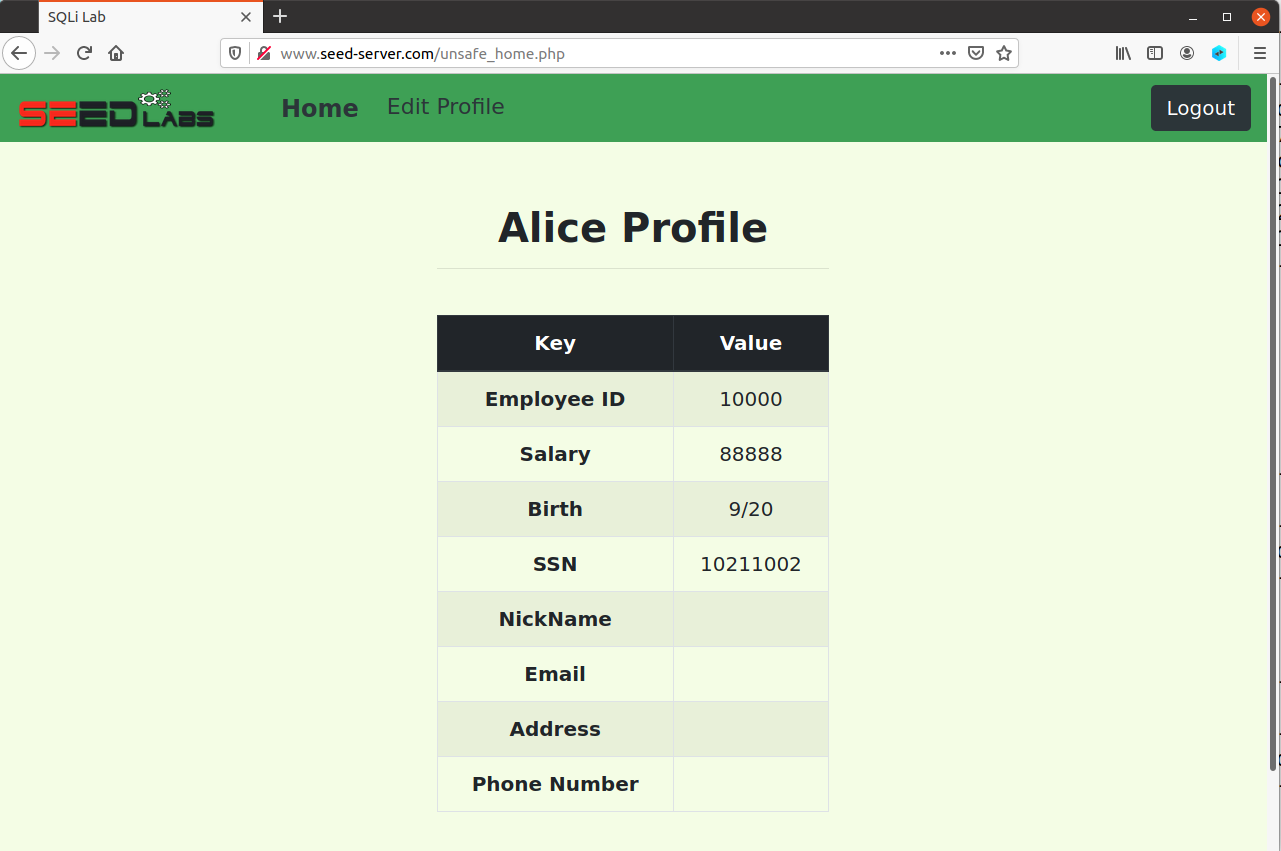
**But searching, i notice that the SQ; injunction does not work against MySQL, because in PHP;s mysqli extension the mysqli::query API does not allot multiple queries to run in the database server.**

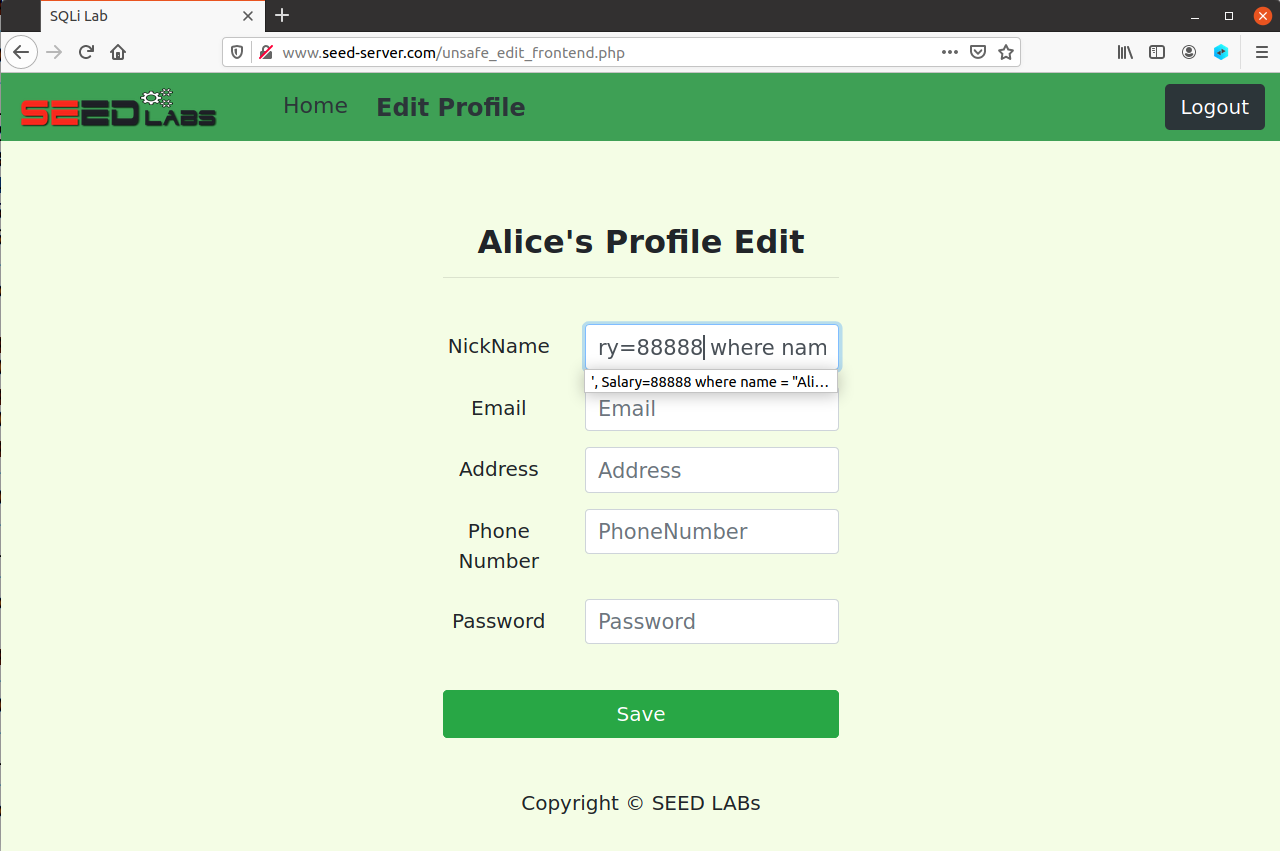
****

Task 3: SQL Injection Attack on UPDATE Statement

Task 3.1: Modify your own salary

**Answer: ', Salary=88888 where name = “Alice” #**  
**As the command written upon, i would use the edit profile nickname part to increase Alice’s salary. I have already know from the code view, the SQL use Update command to update all information. The Update command is Update credential SET Salary=10000 where name=”Alice”. I want increase Alice’s salary so i follow the format of update command and add ‘# to make sure php will run it.**

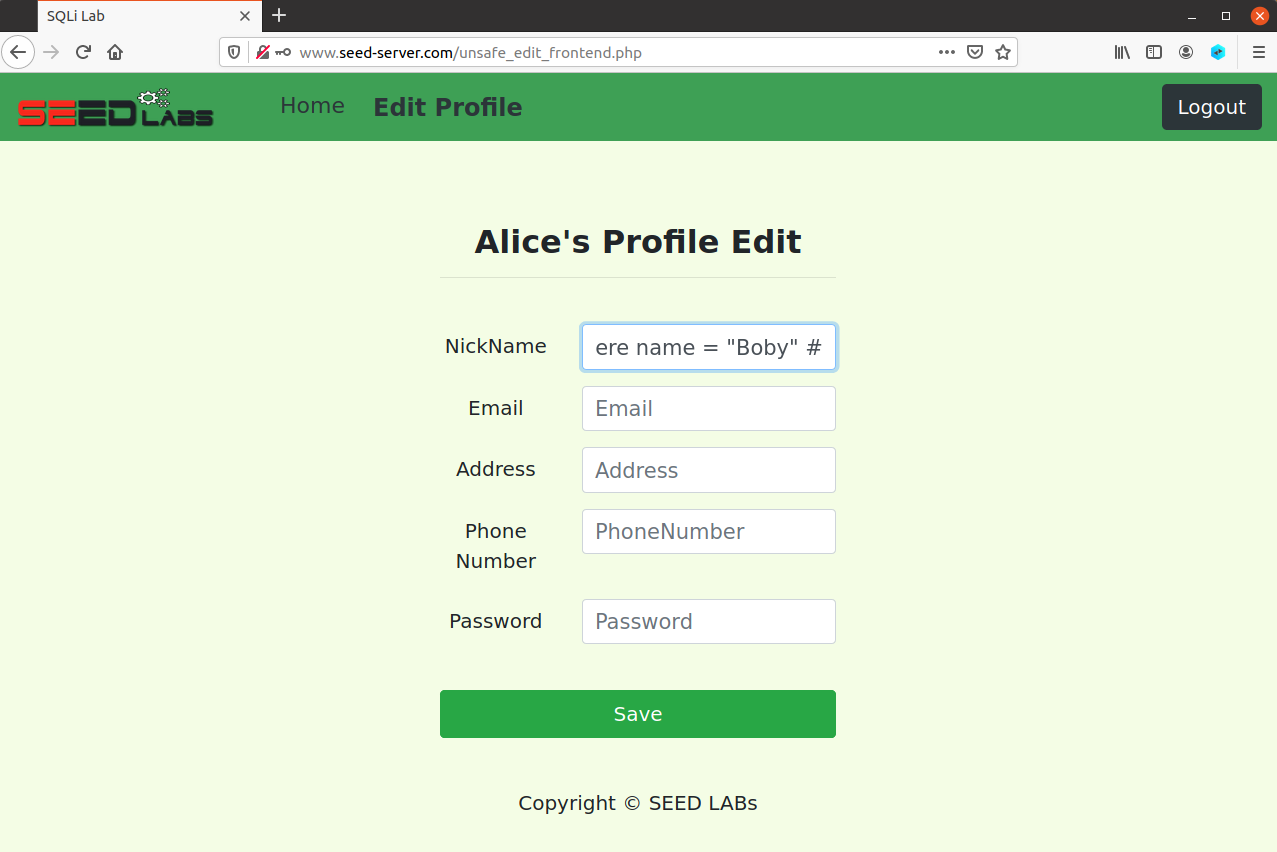


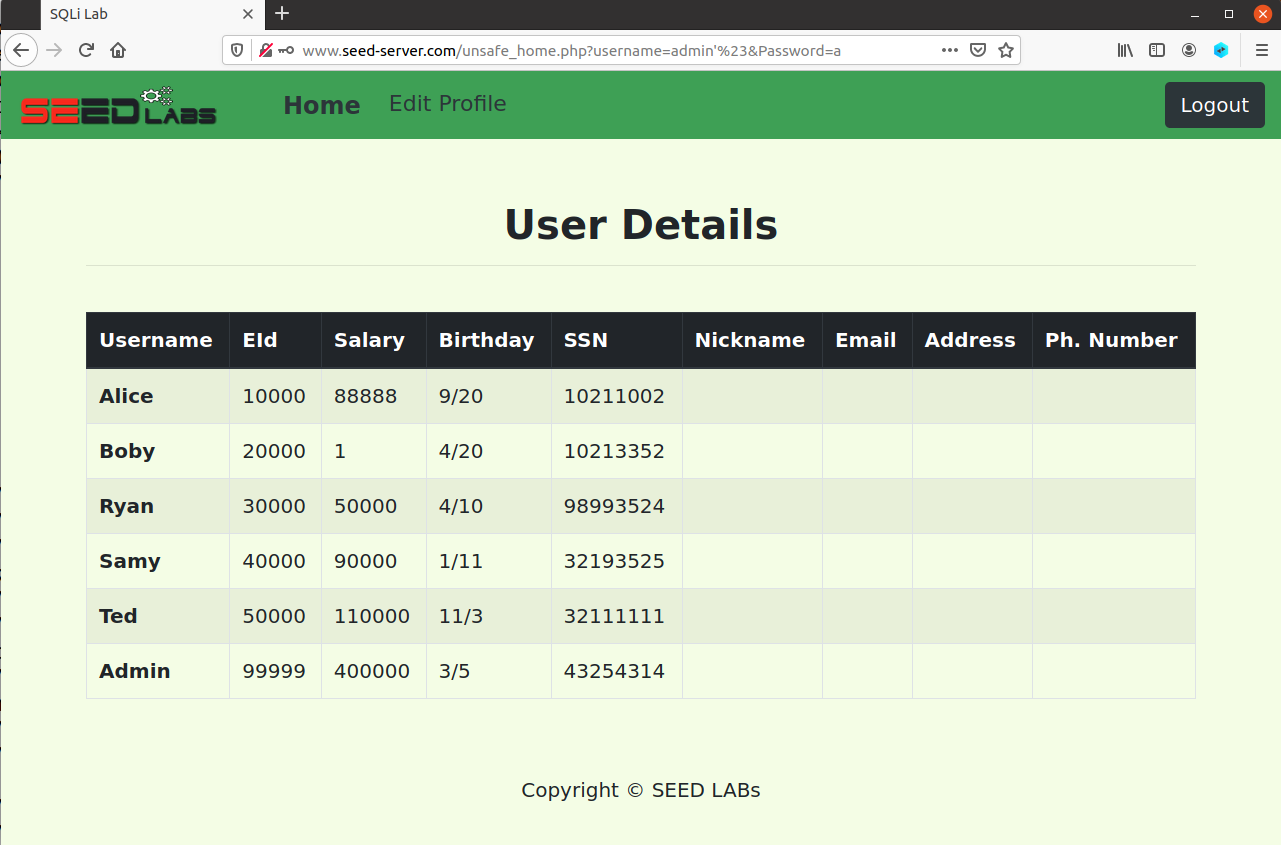


Task 3.2: Modify other people’ salary

**Answer: ', Salary=1 where name = “Boby” #**

**This is uasy to accomplish cause the basic logic is the same.**

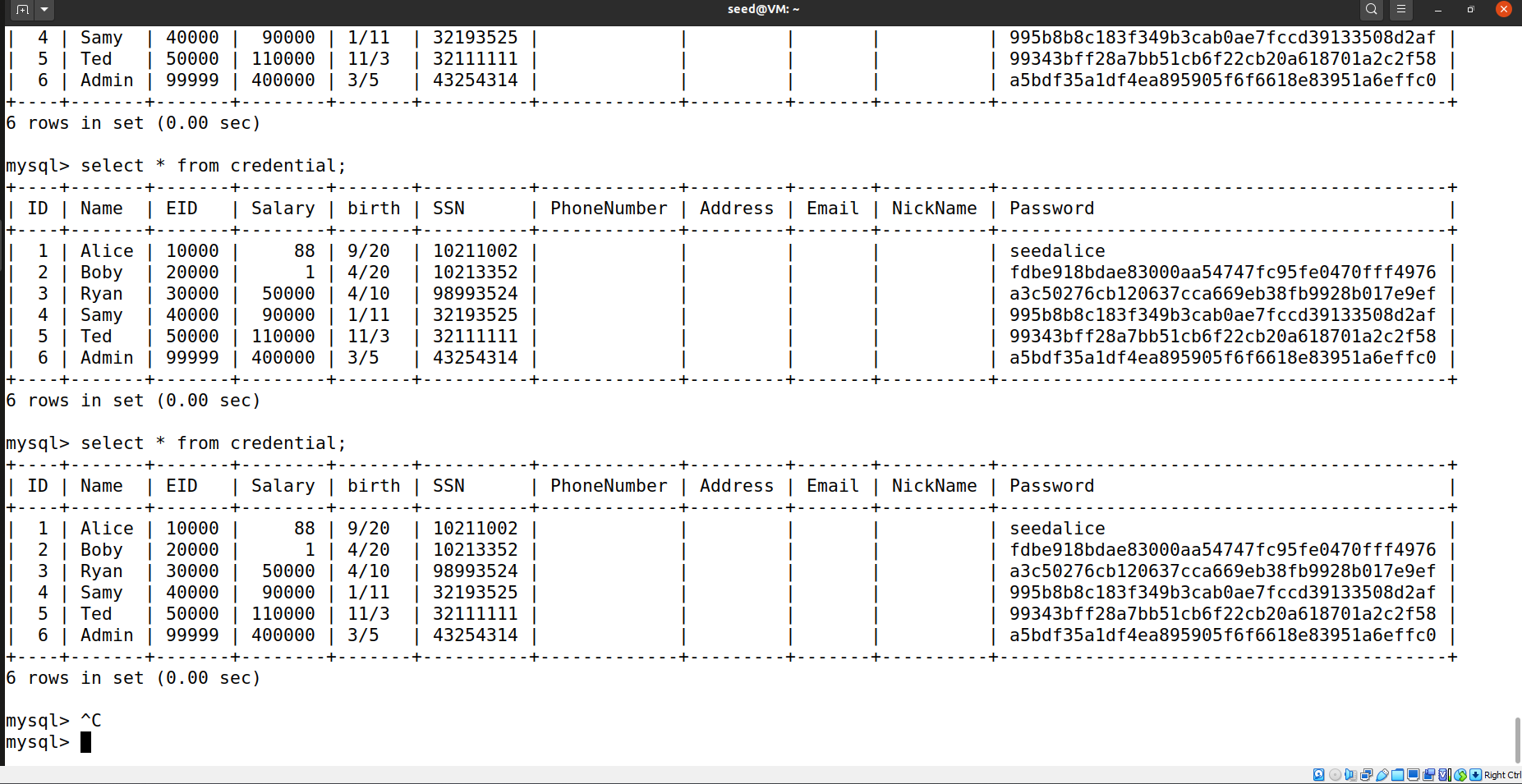
****

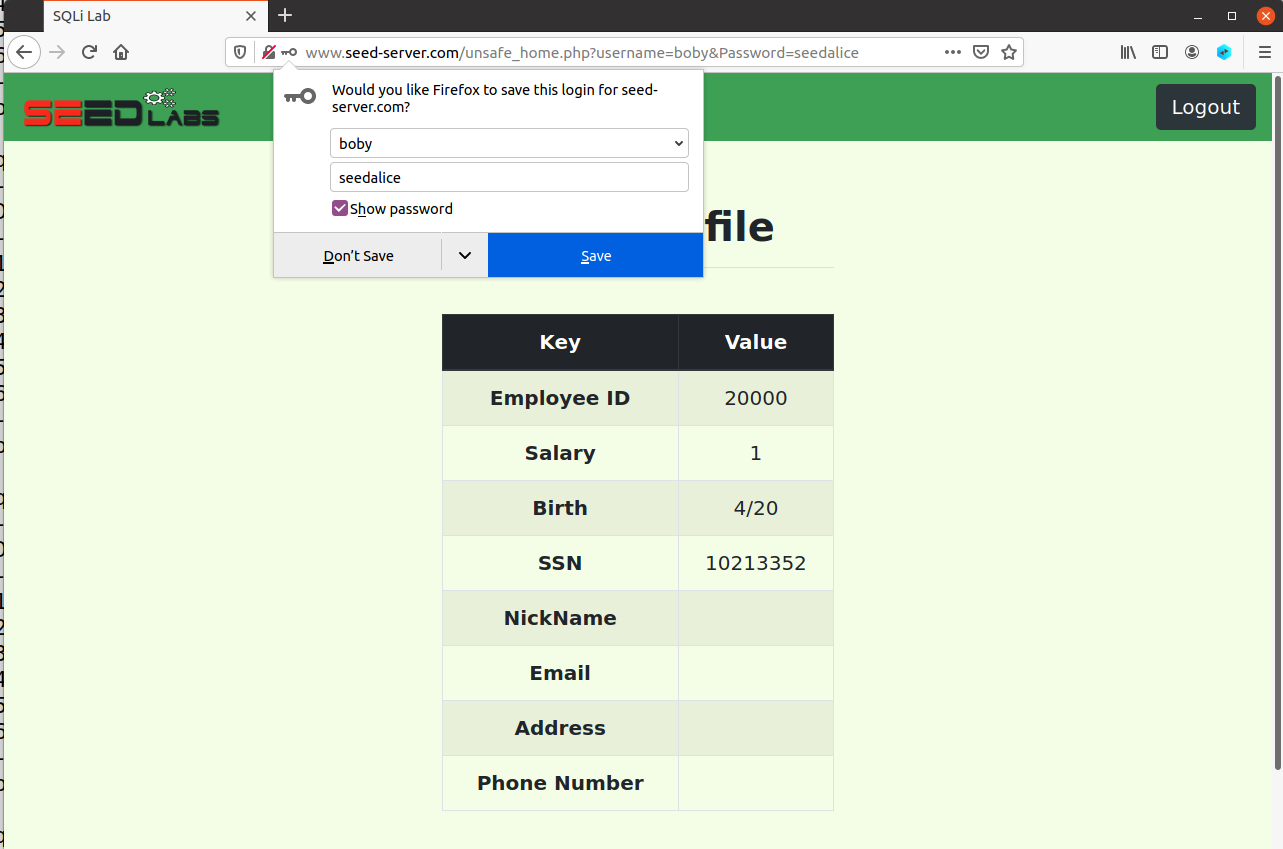


Modify other people’ password.

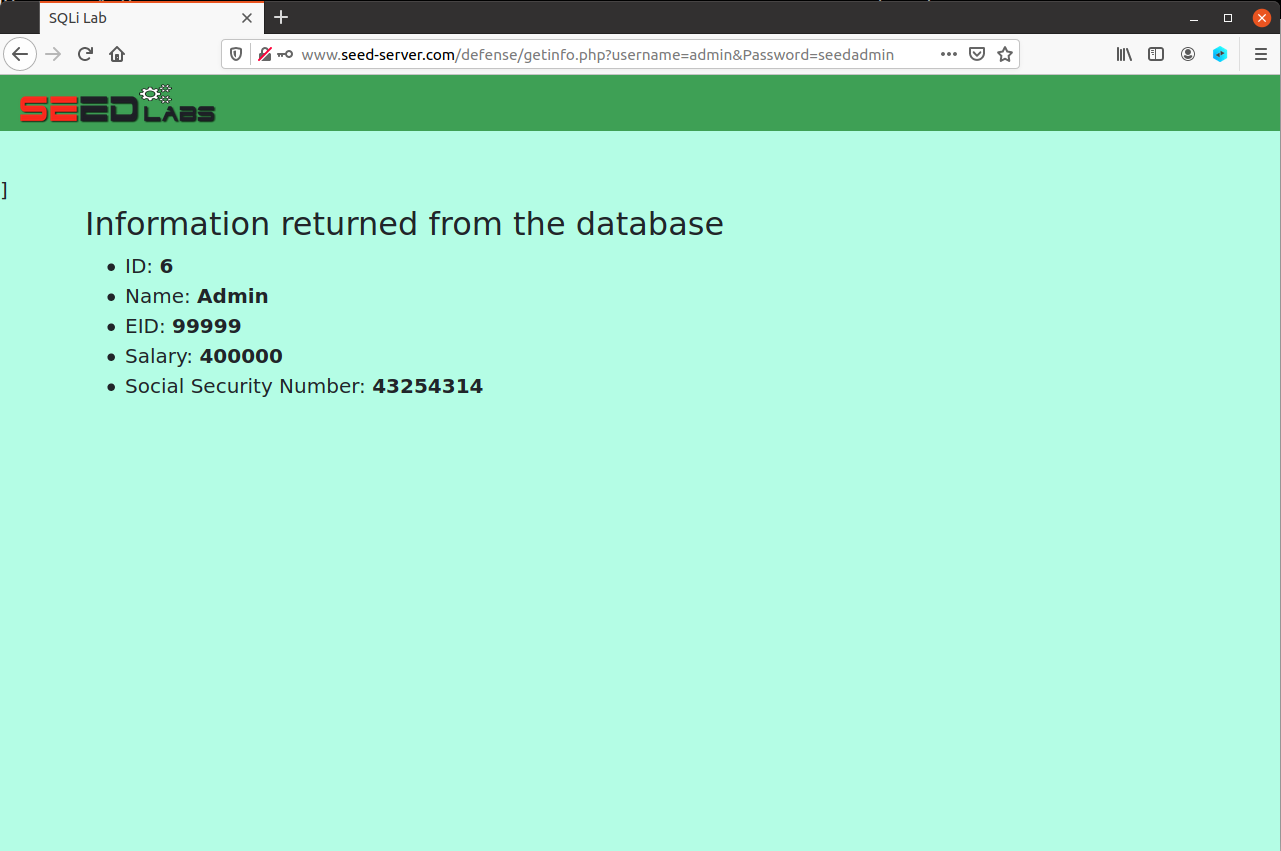
since the data stored the hash value of password, we can not directly set the password by typing **', Password=seedalice where name = “Boby” #. Instead we need to first get the hashed value of “seedalice” which is fdbe918bdae83000aa54747fc95fe0470fff4976. Then we can modify the code into ', Password=’fdbe918bdae83000aa54747fc95fe0470fff4976’ where name = “Boby” #**

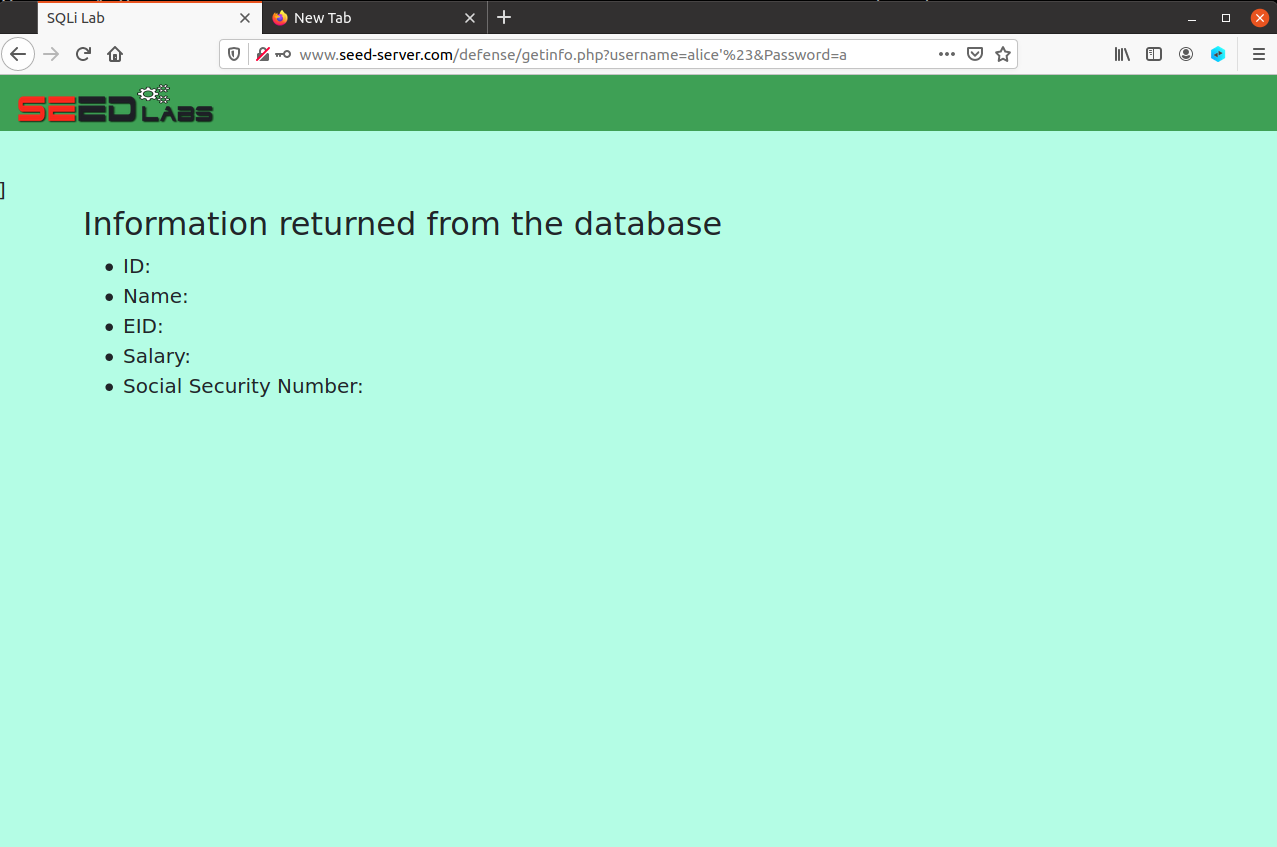
**This follows the rule of change password in the unsafe\_edit\_backend.php. As you can see from database the password is changed.**

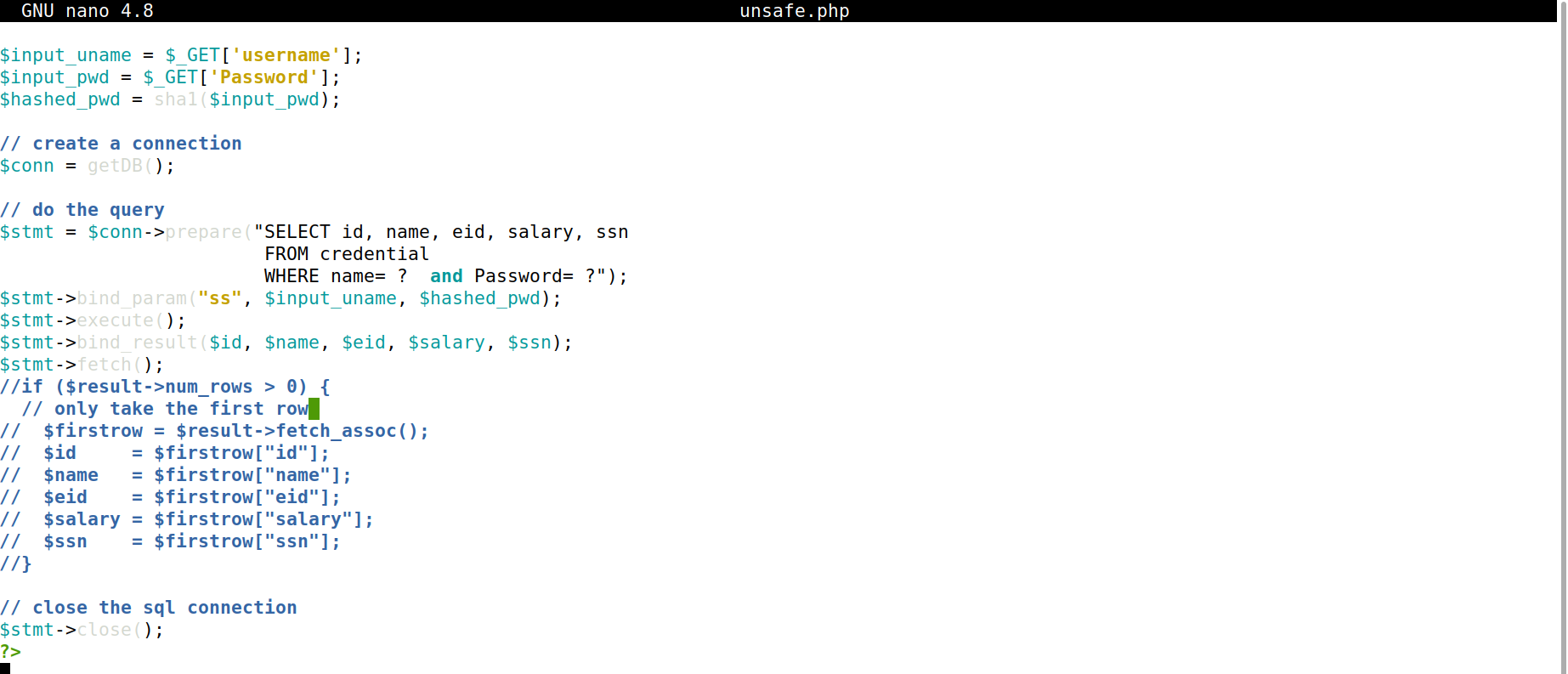




Task 4: Countermeasure — Prepared Statement







**Answer: According to the first figure, we can see that if the user log into the account with the right name password, they will be able to see the information. while in the figure two, when the attacker trying to use SQL attack log into the account, they will see nothing cause in figure three prepare statement limit the type of input name and password into string. Therefore the database can not find a user named alice’# it will then display a empty webpage.**