# FIND YOUR TA

Team Members

Siddhant Bandiwadekar

Pranjal Sondkar

Sanyukta Shandilya

Date Submitted: 5/11/2021

**INDEX**

## OBJECTIVE

To provide a platform for Professors and students to apply and be matched for Teaching Assistant position.

### Current System

The usual process which is currently in place at Syracuse University and well as majority of the Universities across United States is that students apply for various positions through a portal, in our case Handshake. The application of one teaching assistant position is from numerous students with various departments and courses. This makes it difficult and time consuming for Professors to go through all the applications and filter out the student in their relevant course.

Assistantships are highly competitive and landing teaching assistantship is difficult. Landing a TA position is hard for us even more because it can only happen through emails or university application portals which is hard for students because most of the times emails go unread or lost with emails from the university and other such emails. Those who wish to apply for one should present themselves in the best possible way to have a fighting chance at landing one. Teaching assistantship positions are not available in every graduate program and at every college. Moreover, we want a job under a professor with whom our interests and major is a match, also someone who can utilize the skills we have, to profit both.

### Our Proposed System

To overcome above difficulties, we made a web application 'Find Your TA' to make the process of apply and searching for a Teaching Assistant seamless and hassle free.

Platform Expectations:

- Professor Sign up Page
- Student Sign up Page
- Professor Login Page (Private Access)
- Matched Records Page

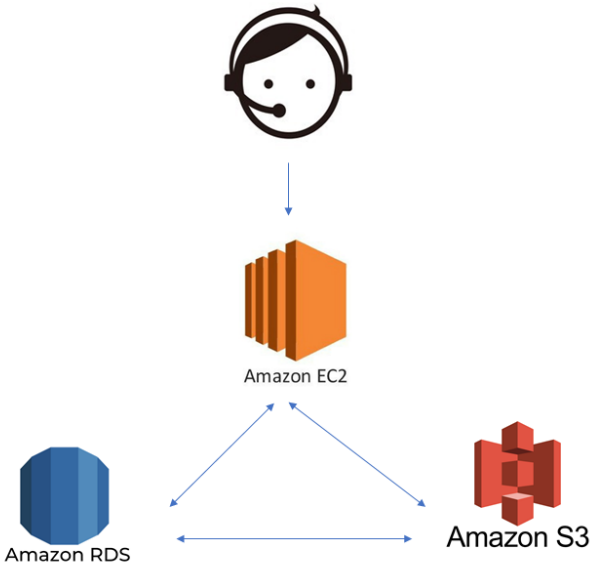
This platform will not require 'Job postings' as such which we see on other common application portals. Here the Professors, along with other details, will just enter the Department and Course for which they are looking a Teaching Assistant. Similarly, students, along with their other details, will enter the department and course for which they want to be considered.

Impact:

It provides an easy way for students to apply without going through numerous job postings and saves time of apply to a lot of positions.

Professor can directly access details of students who have been matched according to the inputs they entered in the form, saving ample time and effort.

## CONFIGURATION OF THE CLOUD SERVICES USED

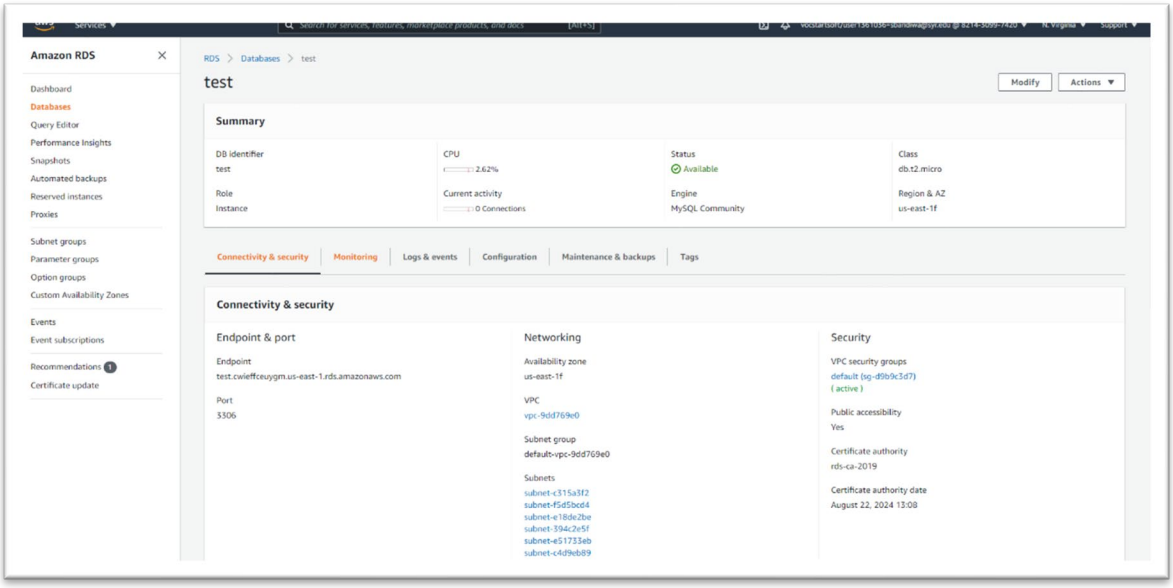



### Amazon RDS

For our project we have created a MySQL database using Amazon RDS. We created a database in the RDS service from the dashboard using the following options: -

- Standard Database Create Method – In this option we do not create the database using automatic recommended settings.
- Engine Option – MySQL
  - Version (MySQL 8.0.17)
- Template – Free tier (For developing and testing new/existing applications to gain a hands-on experience)
- Default name - *test* was used as an identifier for our FA Assistance database server.
- Username and Password credentials were set for accessing the server securely.
- Database Instance Size (For meeting the processing power)
  - db.t2.micro
  - 1Vcpu
  - 1 Gib RAM
- Storage
  - Type – General Purpose (SSD)
  - Allocated Storage – 20 GiB
  - Storage Autoscaling was enabled so that we do not encounter issues if there is excess load on our server.
- Network – Our database is deployed in the default VPC which defines the networking environment for our database.

- We have kept our database publicly accessible so that Amazon EC2 instances and devices outside the VPC and connect to our database.
- Default Security Group was selected to ensure the traffic from the instances.
- Database Port – 3306 (TCP/IP)
- *VPC security groups - default (sg-d9b9c3d7)*
- End Point - test.cwieffceuygm.us-east-1.rds.amazonaws.com

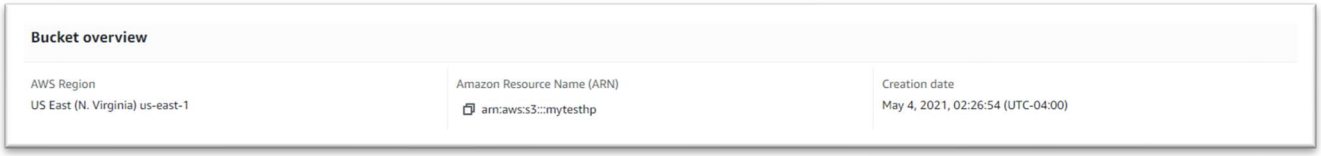



By using the Amazon RDS service, we saved a lot of time as we had all the storage and engine options available in AWS. We would have to install a server separately and install the MySQL server manually on our machine in the traditional approach. RDS also supports scaling of the databases in order to handle increased load. It also has backup and recovery services enabled.

**Amazon S3**

We created a S3 bucket for our project to store our data using the default settings.

- Name – mytesthp
- Zone - US East (N. Virginia) us-east-1
- Link - arn:aws:s3:::mytesthp



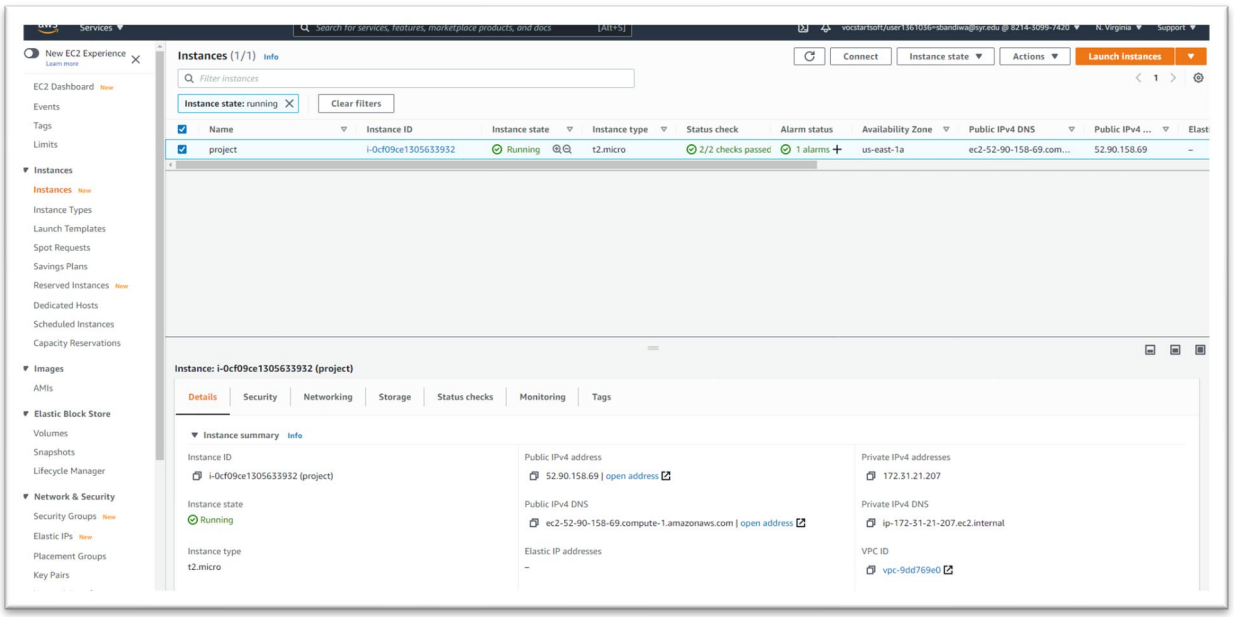

- Upload To S3
  - ✓ aws s3 cp New.csv s3://mytesthp
- Download From S3
  - ✓ aws s3 cp s3://mytesthp/index1.html /home/ubuntu/ProjectPython.py

With a user-friendly web interface Amazon S3 that does all the heavy lifting for us in terms of data security, storage optimization, and data management. It also enables us to customize request metrics and storage class analysis with a variety of filters in order to gain a deeper understanding of our storage. Through Amazon S3, you just pay for the data you use, and it is ensured that our data is not accessed by unauthorized parties.

**Amazon EC2**

- We created an EC2 instance using the following options: -
- Name – Project
- Zone - us-east-1a
- Public IPv4 - 52.90.158.69
- Private IPv4 - 172.31.21.207
- Using the t2. Micro instant type with 1 x vCPU and 1.0Gib RAM.
- Added the storage values (8gb)
- Added a tag with key set to “Name” and Value set to Web Server.
- Created a Security Group which enabled HTTPD from the beginning.
- Used the default security group along with the custom security group.
- We added an HTTP inbound rule which allowed the non-local traffic to access our EC2 instance by creating a new entry for both IPv4 and IPv6 addresses with source set to “anywhere”. By default, all the non-local traffic is blocked.
- Ubuntu Server 20.04 LTS (HVM) with SSD Volume Type was selected.
- IAM Role – awslive (custom created)
- A connection to the server was established using the public/private key pair.

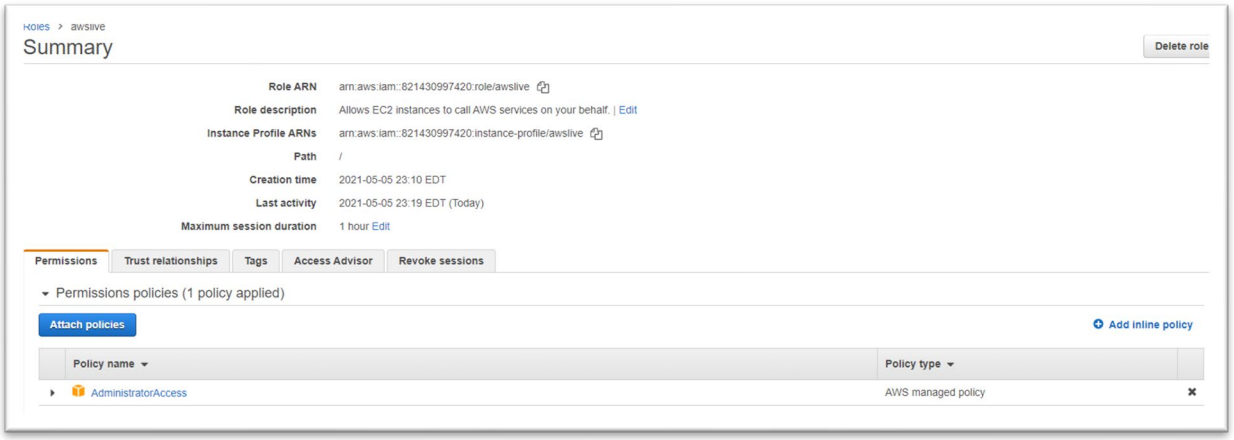
EC2 is cost-effective because it enables users to customize plans to meet their specific needs. This allows us to save money and make the most of our energy. RDS, S3, and IAM are all supported. This is a full-featured computing, processing, and storage system. Security groups on Amazon EC2 serve as virtual firewalls, controlling traffic to one or more instances. Migrating to an EC2 instance is much easier now that users can choose their preferred operating system.




## Identity and Access Management (IAM)

- An IAM Role “awslive” was created.
- Trusting Entities – ec2 Instance.
- Description - Allows EC2 instances to call AWS services on your behalf.
- Selected Policy - Administrator Access

In the AWS console, an IAM function is a collection of permissions that determine what actions are permitted and denied by an individual. It is similar to a user in that any form of person may access it (an individual or AWS service). Role permissions are only used for a short period of time.



## TASKS COMPLETED

### Tasks Completed On EC2 Engine

- All the packages were downloaded from the repositories and then updated using the sudo apt-get update command.
- We installed the MySQL client on our ubuntu server so that we can connect to our RDS server using the following command: -
  - mysql -h test.cwieffceuygm.us-east-1.rds.amazonaws.com -u admin -p
    Password: password
- Primary Database was created - university
- We created **Two** tables in our database with the following attributes: -
  - TABLE – **PROFESSOR**
  - Attributes –
    - Emp ID (INT)
    - First Name (CHAR)
    - Last Name (CHAR)
    - Age (INT)
    - Email (VARCHAR)
    - Phone No (INT)
    - Department (CHAR)
    - Course (CHAR)
  - TABLE – **STUDENT**
  - Attributes –
    - SU ID (INT)
    - First Name (CHAR)
    - Last Name (CHAR)
    - Email (VARCHAR)
    - Degree (CHAR)
    - GPA (INT)
    - Semester (INT)
    - Course (CHAR)
- Values were inserted in both the tables just to make sure that the records are being saved and displayed.
- Values from the above pages were directly inserted in the Professor and Student tables which were directly linked to our RDS instance using php files attached to each webpage.
- The records from these tables were exported to two different .csv files using: -
  - mysql -h test.cwieffceuygm.us-east-1.rds.amazonaws.com -u admin -D university -p --batch --quick -e "SELECT * FROM Professor " | sed 's/\t/","/g;s/^/"/;s/$/"/;s/\n//g' > Prof.csv
  - mysql -h test.cwieffceuygm.us-east-1.rds.amazonaws.com -u admin -D university -p --batch --quick -e "SELECT * FROM Student " | sed 's/\t/","/g;s/^/"/;s/$/"/;s/\n//g' > Stud.csv

- sed 's/\t/","/g;s/^/"/;s/$/"/;s/\n//g' was used to separate the column names in the newly created .csv files.
- "awscli" was installed for communicating with our S3 bucket – mytesthp
- Pandas library was installed for data manipulation and analysis in python.
- PIP was installed to manage and download software packages.
- Fuzzymatcher library was installed using: -
    - sudo pip3 install fuzzymatcher

**Matching Records Using Fuzzymatcher**

- It is a Python package that allows two pandas dataframes to bematched based on one or more common fields. To find possible matches, Fuzzymatcher uses sqlite3's Full Text Search. It then scores match using probabilistic record linkage. Finally, it generates a list of the matches it has discovered, along with their corresponding scores.
- We need to determine the columns to fit for the left and right DataFrames since the columns of courses in both tables have different names. Our Professor will be the left DataFrame in this scenario, while Student details will be on the right.
- Fuzzymatcher uses the fuzzy left join operation and try to find matches and decides the best match for each combination. The matched results DataFrame includes all of the data that has been connected together, as well as a score that indicates the link's consistency. Overall, for medium-sized data sets, fuzzymatcher is a valuable tool to have.
- This matching code was saved as ProjectPython.py and it generated the output csv file containing the matched records.
- The generated csv file containing the matched records was embedded in the Matched Records Page using jquery ajax method.

**Apache**

- On Linux systems, Apache is the most widely used Web server. Clients usually use Web browser software to request and access Web sites.
- Configuration –
    - Sudo apt install apache2
- This creates a directory /www/html in the /var directoy.
- All the files that are to be hosted are stores in the /var/www/html directory.
- By default, the permission to copy files to this directory are restricted. The permissions were enabled using: -
    - sudo chmod 777 /var/www/html -R
- php-mysql was installed to load the php software packages.
- mysqli_connect was used to connect the front end to our RDS backend by passing the RDS Endpoing, database instance username, password, and database name.
- The website was then collectively moved from S3 to /var/www/html for hosting

Following pages were created using HTML, CSS and JavaScript. Used Python and Php in the backend to connect them to the 'university' database on RDS MYSQL server and perform match logic.

**Created Sign up Pages**

i) Professor Sign Up Page

This is the entry page or first page of our application. This meant for Professors to sign up on the portal by entering their details.
This page consists of the following fields,

- First Name – This field is captured and saved as a character field in the database table
- Last Name – This field is captured and saved as a character field in the database table
- Employee ID – This field is captured and saved as a numeric field in the database table
- Age – This field is captured and saved as a numeric field in the database table
- Email – This field is captured and saved as a character field in the database table
- Phone No - This field is captured and saved as a numeric field in the database table
- Department - This field is captured and saved as a numeric field in the database table
- Course – This field is captured and saved as a numeric field in the database table

After entering theses details and clicking on submit, a webpage appears confirming that your details have been recorded. These details are then stored in 'Professor' table of 'university' database in RDS.
The 'Reset' button clears all the fields to re-enter the information.
The 'Student Sign up' button takes user to student sign up page.

ii) Student Sign Up Page

This page is meant for Students to sign up on the portal by entering their details.
This page consists of the following fields,

- First Name - This field is captured and saved as a character field in the database table
- Last Name - This field is captured and saved as a character field in the database table
- SUID - This field is captured and saved as a numeric field in the database table
- Email - This field is captured and saved as a character field in the database table
- GPA - This field is captured and saved as a numeric field in the database table

- Student Department - This field is captured and saved as a character field in the database table
- Course Name - This field is captured and saved as a character field in the database table

After entering theses details and clicking on submit, a webpage appears confirming that your details have been recorded. These details are then stored in 'Student' table of 'university' database in RDS.
The 'Reset' button clears all the fields to re-enter the information.
The 'Professor Sign up' button takes user back to professor sign up page.

**Created Professor Login Page**

This page is for authentication purpose implemented using AWS IAM. Our idea behind making this page was to keep the Matched Records secure for Professor's use itself. By the use of this login page, students will not be able to access the matched records table that enables Professors to shortlist them.

This page has two fields,

- Email
- Employee ID

After clicking on 'Submit' the code fetches and matches the records in 'Professor' table. Once the details are validated, it shows the Matched Records page. If the code cannot find such data in the database, or if the email or employee ID is incorrectly entered, a pop-up appears denying any further access.

**Created Matched Records Page**

Once the Professor Email and ID is validated, the portal displays this page. It consists of a table displaying the matched records.

The match is done based on department and course. When Professor registers, they mention their department and the course for which they want a teaching assistant. Similarly, when students register, they mention their department and the course for which they prefer working as a teaching assistant.

The logic developed in Python, implemented by Fuzzymatcher, which is explained further in the report, matches the professors with students based on the above mentioned department and course details. For example, a Professor with 'Analytics' course, will be matched with all the students who have mentioned the same as their course.

This way it becomes easy for the Professors to filter out and shortlist students who are already studying or working in their relative domain.

## ISSUES ENCOUNTERED

Establishing connection between RDS and Web page –
Initially, we had created an RDS and our web pages. It was not clear as to how can we make a connection between them such that the data entered by Professor or a Student should be stored in the respective tables in our database Also, since we were performing join on the two tables after the data is entered. Hence, we had to figure out a way to display those matched records, generated on the server, on the Matched Records page.
Apart from that, by default, DB instances don't allow access. Access is granted through a security group associated with the VPC that allows traffic into and out of the DB instance.

Logic and implementation of matching student and professor record –
Our two tables, Professor and Student, do not have a unique identifies as such, like a Primary key. Hence, we had to find out a way or a logic to perform join on these table in order to display the best matched students to professor. This led us to discover the fuzzymatcher package in python.
The only disadvantage of this tool is that fuzzy matching takes time to compute as the number of rows increases and is harder to implement.

## LESSONS LEARNED

Lesson 1 –
We were able to network, have central and secure data access, which facilitates increased collaboration and ultimately improved performance, by using AWS cloud services.

Lesson 2 –
We are able to reduce the total IT infrastructure expenditures with AWS because there is no need to purchase hardware and related applications for an IT infrastructure implementation, resulting in significant cost savings. Furthermore, we just pay for resources and facilities that we use on a subscription basis.

**REFERENCES**

Vohra, D. (2016). Using the Amazon EC2. *Pro Docker,* 229-252. doi:10.1007/978-1-4842-1830-3_15

Gulabani, S. (2017). Practical Amazon EC2, SQS, Kinesis, and S3. doi:10.1007/978-1-4842-2841-8

Shackelford, A. (2015). Getting started with Amazon web services. *Beginning Amazon Web Services with Node.js,* 1-29. doi:10.1007/978-1-4842-0653-9_1

Nadon, J. (2017). Database services in AWS. *Website Hosting and Migration with Amazon Web Services,* 127-151. doi:10.1007/978-1-4842-2589-9_10

Beach, B. (2014). SQL server RDS Parameters. *Pro Powershell for Amazon Web Services,* 279-284. doi:10.1007/978-1-4302-6452-1_17

Goodwill, J. (2002). Integrating the apache http server. *Apache Jakarta Tomcat,* 151-158. doi:10.1007/978-1-4302-0851-8_9

Aishwarya Anand. (2017). Managing infrastructure in Amazon using EC2, CLOUDWATCH, EBS, IAM and CloudFront. *International Journal of Engineering Research and, V6*(03). doi:10.17577/ijertv6is030335

Introducing mysql. (n.d.). *Beginning PHP and MySQL 5,* 573-580. doi:10.1007/978-1-4302-0117-5_24

Amazon web Services (AWS) - cloud computing services. (n.d.). Retrieved May 11, 2021, from https://aws.amazon.com/

Ubuntu documentation. (n.d.). Retrieved May 11, 2021, from https://help.ubuntu.com/

Sabharwal, N., & Wali, P. (2013). *Cloud capacity management*. Berkeley, CA: Apress.

Levitt, J. (2005). *The web developer's guide to Amazon e-commerce service: Developing web applications using Amazon Web Services and PHP*. Austin, TX.: J9T Pub.

Amazon EC2 instances - Amazon Elastic Compute Cloud. (n.d.). Retrieved May 11, 2021, from https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/Instances.html

AWS tutorial: How to host a website on AWS ec2 instance. (2019, July 30). Retrieved May 11, 2021, from https://www.youtube.com/watch?v=hOYg-ClP84g

Brandon, J. (2020, February 01). AWS: Your complete guide to Amazon web services & features. Retrieved May 11, 2021, from https://www.techradar.com/news/aws

Fundamentals of cloud management. (2014). Cloud Management and Security, 29-46. doi:10.1002/9781118817087.ch3

# APPENDIX

## Tasks Screenshots

Connecting To EC2

```
login as: ubuntu
Authenticating with public key "imported-openssh-key"
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.4.0-1045-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

 System information disabled due to load higher than 1.0

1 update can be applied immediately.
To see these additional updates run: apt list --upgradable



The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
```

Updating the ubuntu system

```
ubuntu@ip-172-31-21-207:~$ sudo apt-get update
```

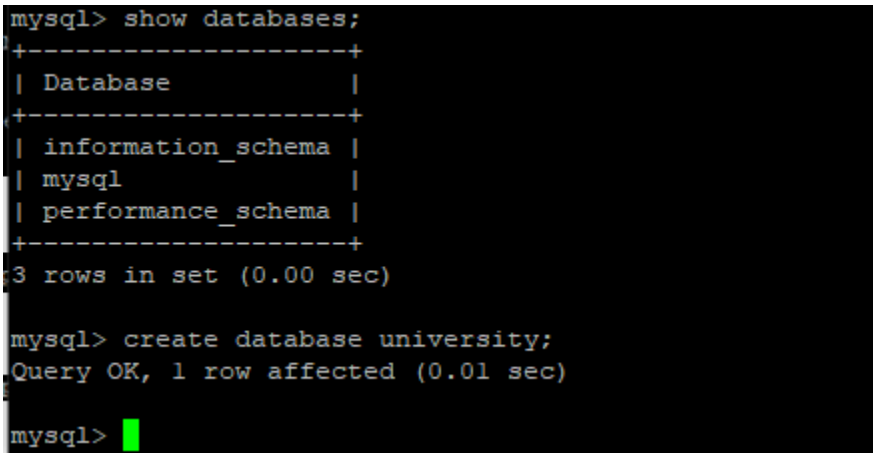
Installing MySql Client

```
ubuntu@ip-172-31-21-207:~$ sudo apt-get install mysql-client
```

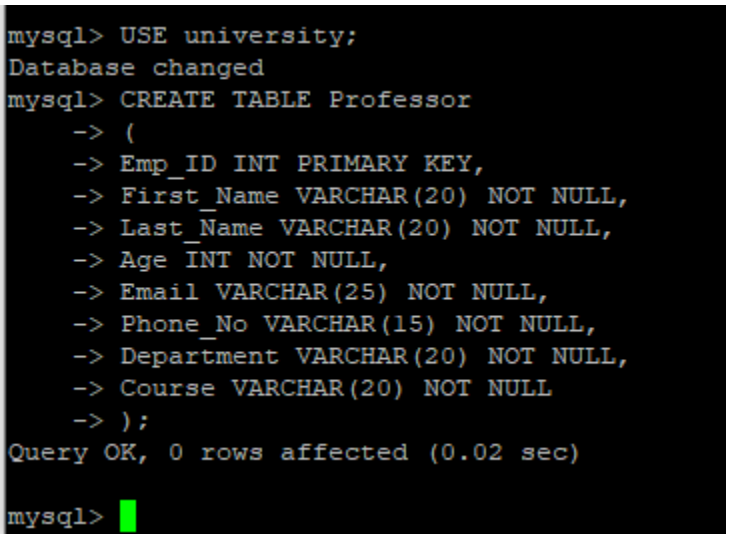
Connecting To Our RDS (MySQL Server)

```
ubuntu@ip-172-31-21-207:~$ mysql -h test.cwieffceuygm.us-east-1.rds.amazonaws.co
m -u admin -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 22
Server version: 8.0.20 Source distribution

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

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>
```

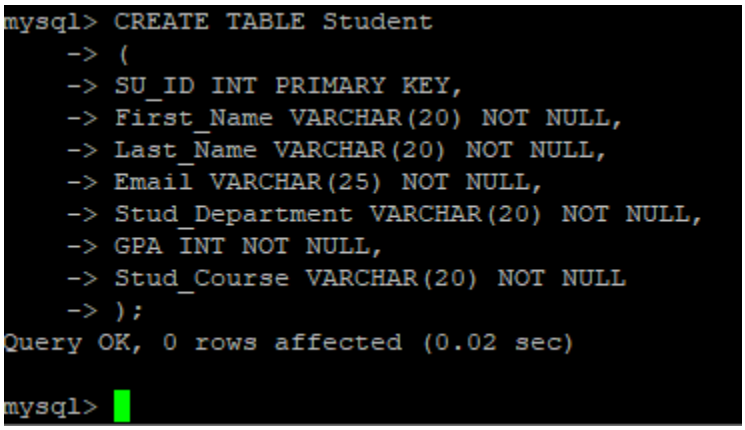
Creating primary database




Using the newly created database and creating Professor Table




Creating Student Table



Showing Tables In The Database

```
mysql> show tables;
+----------------------+
| Tables_in_university |
+----------------------+
| Professor            |
| Student              |
+----------------------+
2 rows in set (0.00 sec)

mysql>
```

Inserting Values In Professor Table and Displaying

```
mysql> select * from Professor;
+--------+------------+--------------+-----+------------------+------------+------------+----------+
| Emp_ID | First_Name | Last_Name    | Age | Email            | Phone_No   | Department | Course   |
+--------+------------+--------------+-----+------------------+------------+------------+----------+
|    111 | Pranjal    | Sondkar      |  24 | pranj@syr.edu    | 2853956611 | DS         | Database |
|    123 | Siddhant   | Bandiwadekar |  24 | sbandiwa@syr.edu | 3153956611 | IM         | Cloud    |
+--------+------------+--------------+-----+------------------+------------+------------+----------+
2 rows in set (0.00 sec)
```

Inserting Values In Student Table and Displaying

```
mysql> Select * from Student;
+-----------+------------+------------+------------------+------------------+------+-------------+
| SU_ID     | First_Name | Last_Name  | Email            | Stud_Department  | GPA  | Stud_Course |
+-----------+------------+------------+------------------+------------------+------+-------------+
| 164845294 | Shyamal    | Parekh     | sparekh@syr.edu  | DS               | 3.80 | Analytics   |
| 364845292 | Sandesh    | Bhat       | sabhat@syr.edu   | ECS              | 3.60 | Cloud       |
| 464845294 | Sahil      | Shah       | sshah@syr.edu    | DS               | 3.70 | Analytics   |
| 564845294 | Bhavya     | Gala       | bgala@syr.edu    | IM               | 3.76 | Cloud       |
| 564848294 | Amey       | Sawant     | aswant@syr.edu   | IM               | 3.00 | Database    |
| 664845294 | Kayomarz   | Buhariwala | kbuhara@syr.edu  | IM               | 3.31 | Database    |
| 764845294 | Dhaval     | Sonavaria  | dhsonava@syr.edu | IM               | 3.20 | Analytics   |
+-----------+------------+------------+------------------+------------------+------+-------------+
7 rows in set (0.00 sec)
```

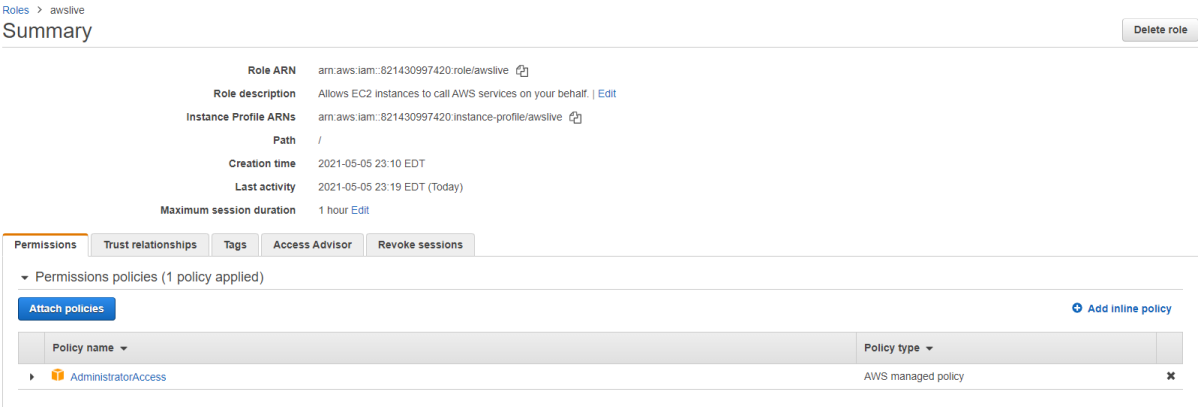
Displaying and Verifying Matching Students Using Inner Join for Courses in Both Tables

```
mysql> SELECT
    -> Professor.First_Name, Professor.Last_Name, Student.First_Name, Student.Last_Name, Student.Email, Student.GPA,Student.Stud_Course
    -> FROM Professor
    -> INNER JOIN Student ON Professor.Course = Student.Stud_Course;
+------------+--------------+------------+-----------+----------------+------+-------------+
| First_Name | Last_Name    | First_Name | Last_Name | Email          | GPA  | Stud_Course |
+------------+--------------+------------+-----------+----------------+------+-------------+
| Siddhant   | Bandiwadekar | Sandesh    | Bhat      | sabhat@syr.edu | 3.60 | Cloud       |
| Siddhant   | Bandiwadekar | Bhavya     | Gala      | bgala@syr.edu  | 3.76 | Cloud       |
| Pranjal    | Sondkar      | Amey       | Sawant    | aswant@syr.edu | 3.00 | Database    |
| Pranjal    | Sondkar      | Kayomarz   | Buhariwala| kbuhara@syr.edu| 3.31 | Database    |
+------------+--------------+------------+-----------+----------------+------+-------------+
4 rows in set (0.00 sec)

mysql>
```

Exporting .csv Files

```
ubuntu@ip-172-31-21-207:~$ mysql -h test.cwieffceuygm.us-east-1.rds.amazonaws.com -u admin -D university -p --batch --quick -e "SELECT * FROM Professor
 " | sed 's/\t/","/g;s/^/"/;s/$/"/;s/\n//g' > Prof.csv
Enter password:
ubuntu@ip-172-31-21-207:~$ mysql -h test.cwieffceuygm.us-east-1.rds.amazonaws.com -u admin -D university -p --batch --quick -e "SELECT * FROM Student "
 | sed 's/\t/","/g;s/^/"/;s/$/"/;s/\n//g' > Stud.csv
```

Creating IAM Role With Administrative Access




Installing awscli For S3

```
ubuntu@ip-172-31-21-207:~$ sudo apt-get install awscli
```

Uploading Files on S3

```
ubuntu@ip-172-31-21-207:~$ aws s3 cp Prof.csv s3://mytesthp
upload: ./Prof.csv to s3://mytesthp/Prof.csv
ubuntu@ip-172-31-21-207:~$ aws s3 cp Stud.csv s3://mytesthp
upload: ./Stud.csv to s3://mytesthp/Stud.csv
```

Installing Pandas

```
ubuntu@ip-172-31-21-207:~$ sudo apt-get install python3-pandas
```

Installing pip

```
ubuntu@ip-172-31-21-207:~$ sudo apt install python3-pip
```

Installing Fuzzymatcher (Python)

```
ubuntu@ip-172-31-21-207:~$ sudo pip3 install fuzzymatcher
```

Installing Apache Server For Hosting Website

```
ubuntu@ip-172-31-21-207:/var$ sudo apt install apache2
```

Allowing Access To /var/www/html folder

```
ubuntu@ip-172-31-21-207:~$ sudo chmod 777 /var/www/html -R
```

Installing Php-Mysql

```
ubuntu@ip-172-31-21-207:/var/www/html$ sudo apt install php libapache2-mod-php php-mysql
```

Moving The Webpages From S3 To /var/www/html For Hosting

```
ubuntu@ip-172-31-21-207:/var/www/html$ ls
Success.html  Success1.html  index.html  php_insert.php  php_insert1.php  style.css  task.html  task1.html
ubuntu@ip-172-31-21-207:/var/www/html$
```

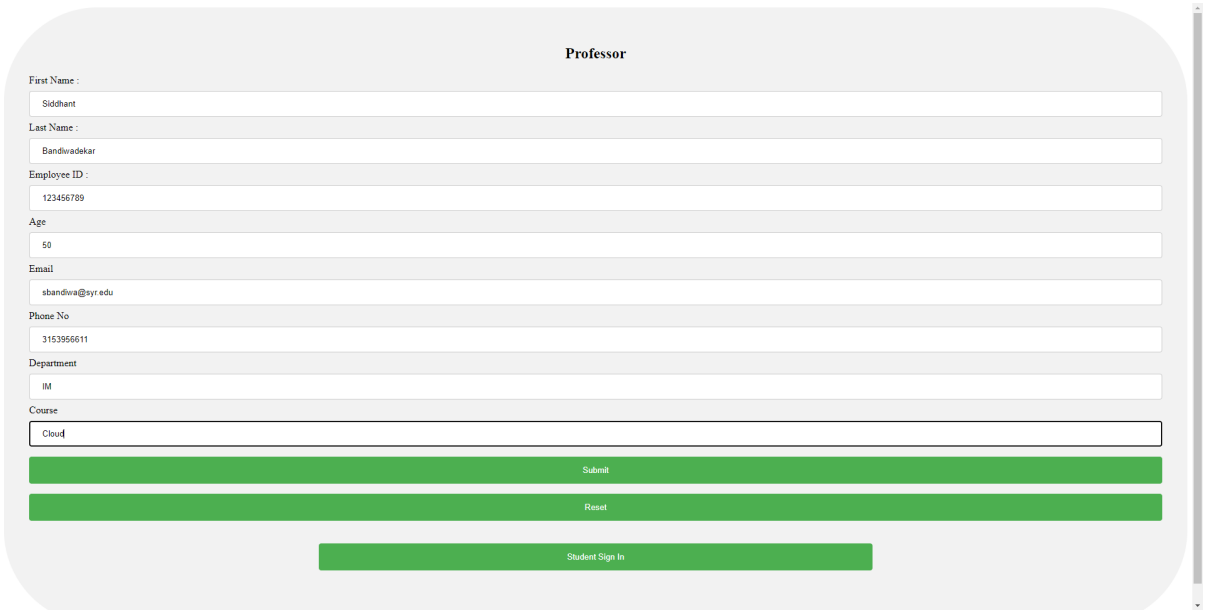
Restarting and Checking The Status Of Apache Server

```
ubuntu@ip-172-31-21-207:/var/www/html$ sudo systemctl restart apache2
ubuntu@ip-172-31-21-207:/var/www/html$ sudo systemctl status apache2
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)
     Active: active (running) since Sun 2021-05-09 07:39:22 UTC; 5s ago
       Docs: https://httpd.apache.org/docs/2.4/
    Process: 10866 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
   Main PID: 10879 (apache2)
      Tasks: 6 (limit: 1160)
     Memory: 10.1M
     CGroup: /system.slice/apache2.service
             ├─10879 /usr/sbin/apache2 -k start
             ├─10880 /usr/sbin/apache2 -k start
             ├─10881 /usr/sbin/apache2 -k start
             ├─10882 /usr/sbin/apache2 -k start
             ├─10883 /usr/sbin/apache2 -k start
             └─10884 /usr/sbin/apache2 -k start

May 09 07:39:22 ip-172-31-21-207 systemd[1]: apache2.service: Succeeded.
May 09 07:39:22 ip-172-31-21-207 systemd[1]: Stopped The Apache HTTP Server.
May 09 07:39:22 ip-172-31-21-207 systemd[1]: Starting The Apache HTTP Server...
May 09 07:39:22 ip-172-31-21-207 systemd[1]: Started The Apache HTTP Server.
```

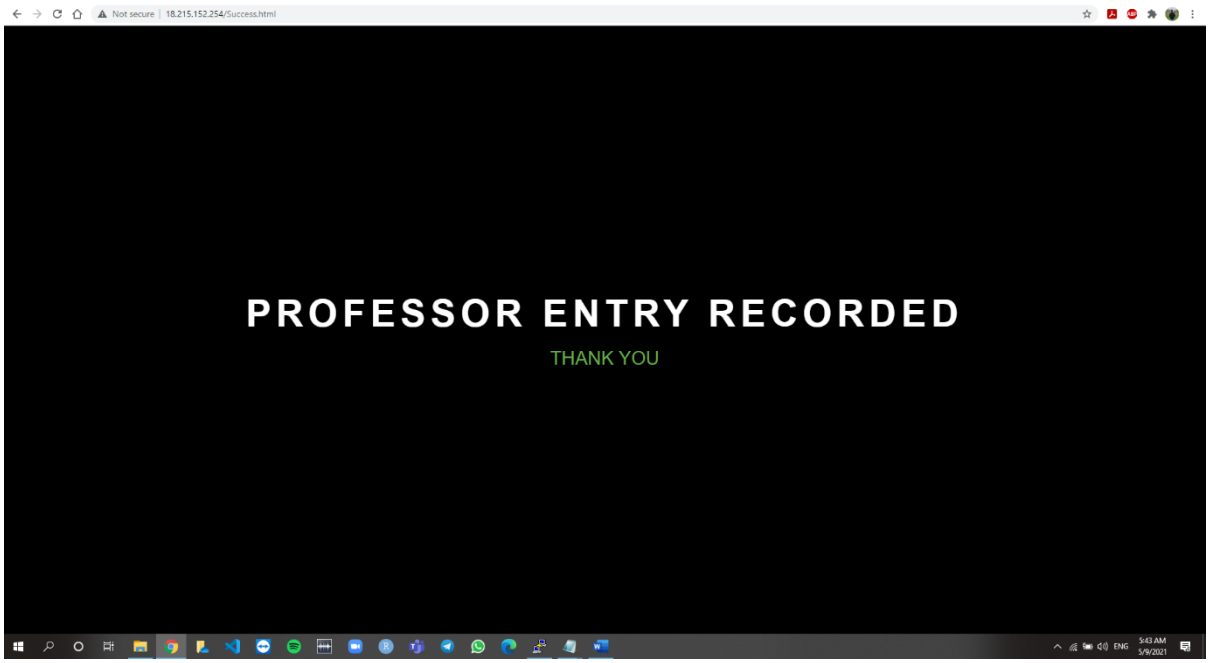
Establishing Connection With The RDS Server With The Webpage: -

$connect = mysqli_connect('test.cwieffceuygm.us-east-1.rds.amazonaws.comm', 'admin', 'password','university');

**Professor Webpage**



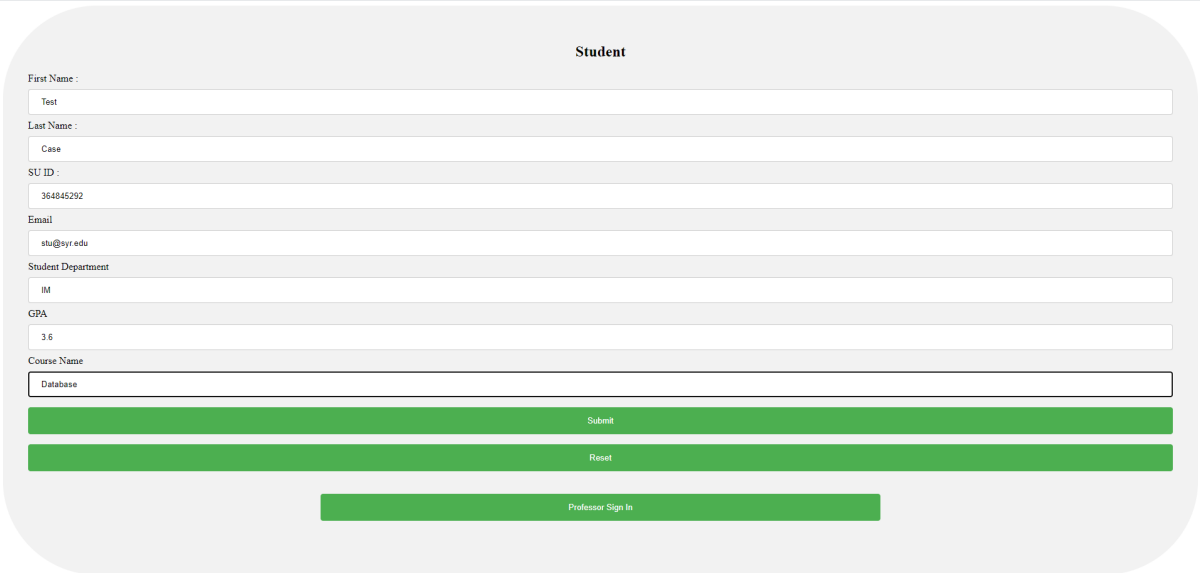
Success Message



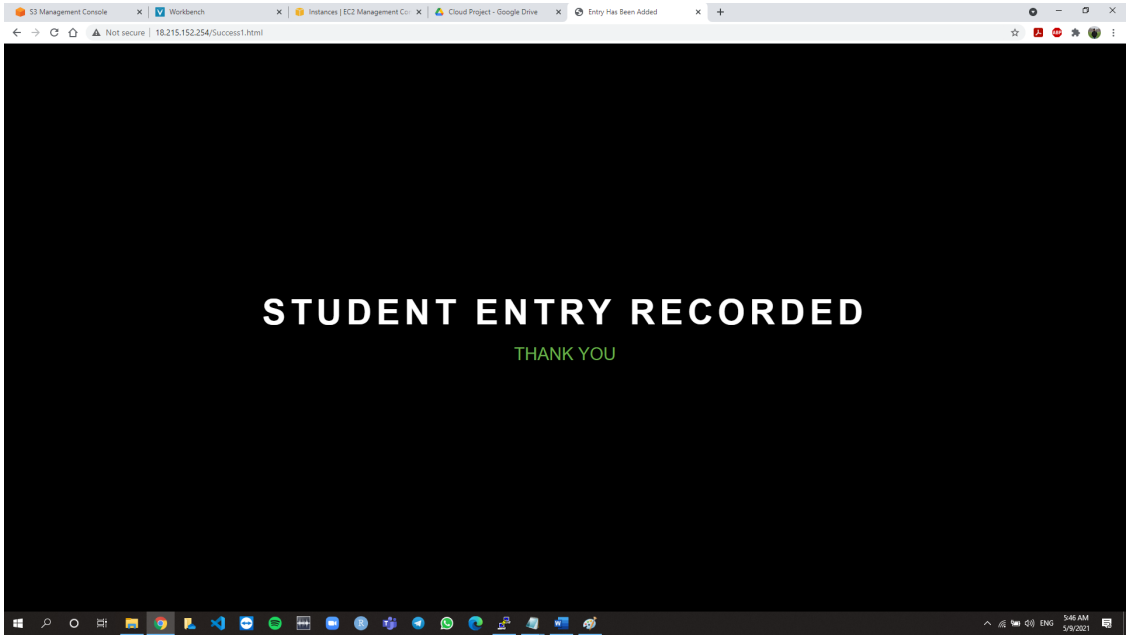

Database Update (Table Name – Professor)

```
mysql> select * from Professor;
+-----------+------------+--------------+-----+------------------+------------+------------+---------+
| Emp_ID    | First_Name | Last_Name    | Age | Email            | Phone_No   | Department | Course  |
+-----------+------------+--------------+-----+------------------+------------+------------+---------+
| 123456789 | Siddhant   | Bandiwadekar |  50 | sbandiwa@syr.edu | 3153956611 | IM         | Cloud   |
+-----------+------------+--------------+-----+------------------+------------+------------+---------+
1 row in set (0.00 sec)

mysql>
```

**Student Webpage**



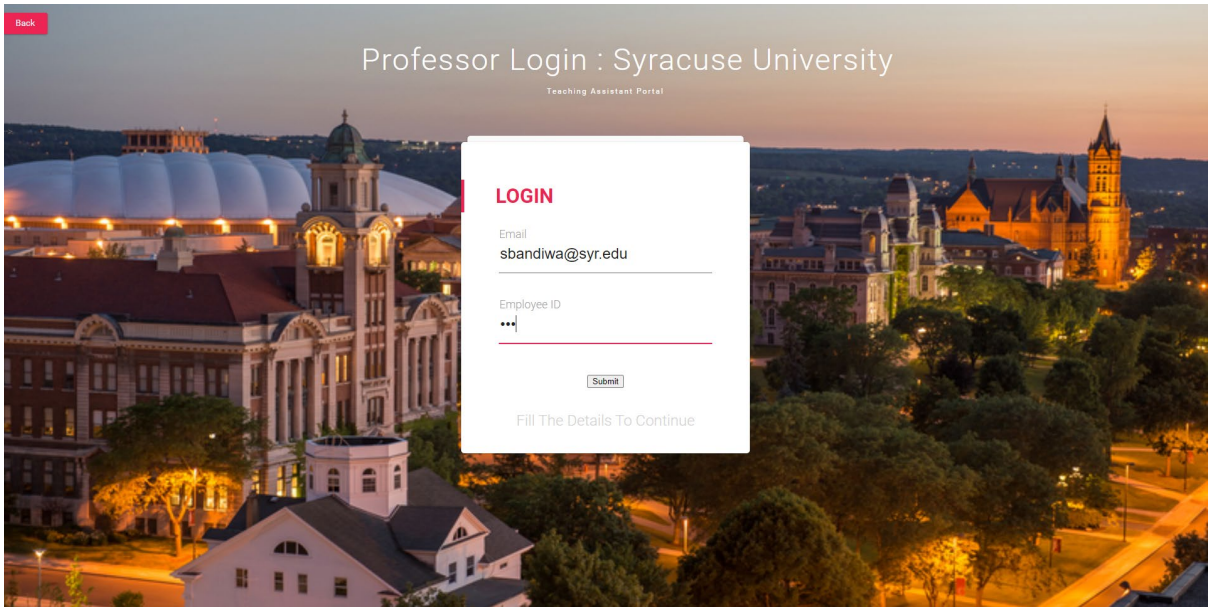
Success Message After Updating Tables



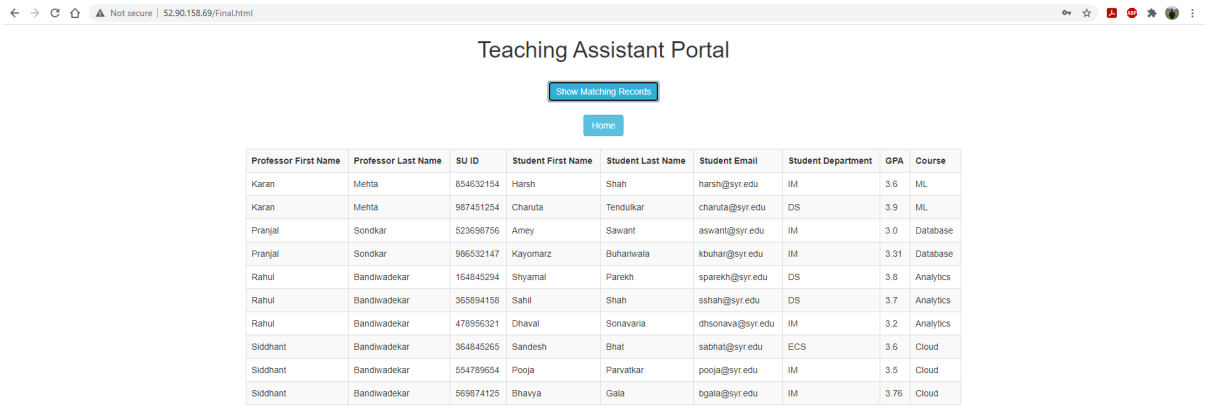

Database Update (Table Name – Student)

```
mysql> select * from Student
    -> ;
+-----------+------------+-----------+-------------+------------------+------+--------------+
| SU_ID     | First_Name | Last_Name | Email       | Stud_Department  | GPA  | Stud_Course  |
+-----------+------------+-----------+-------------+------------------+------+--------------+
| 364845292 | Test       | Case      | stu@syr.edu | IM               | 3.60 | Database     |
+-----------+------------+-----------+-------------+------------------+------+--------------+
1 row in set (0.00 sec)

mysql>
```

Professor Private Access



Matching Records




| Professor First Name | Professor Last Name | SU ID | Student First Name | Student Last Name | Student Email | Student Department | GPA | Course |
|---|---|---|---|---|---|---|---|---|
| Karan | Mehta | 854632154 | Harsh | Shah | harsh@syr.edu | IM | 3.6 | ML |
| Karan | Mehta | 987451254 | Charuta | Tendulkar | charuta@syr.edu | DS | 3.9 | ML |
| Pranjal | Sondkar | 523698756 | Amey | Sawant | aswant@syr.edu | IM | 3.0 | Database |
| Pranjal | Sondkar | 986532147 | Kayomarz | Buhariwala | kbuhar@syr.edu | IM | 3.31 | Database |
| Rahul | Bandiwadekar | 164845294 | Shyamal | Parekh | sparekh@syr.edu | DS | 3.8 | Analytics |
| Rahul | Bandiwadekar | 365894158 | Sahil | Shah | sshah@syr.edu | DS | 3.7 | Analytics |
| Rahul | Bandiwadekar | 478956321 | Dhaval | Sonavaria | dhsonava@syr.edu | IM | 3.2 | Analytics |
| Siddhant | Bandiwadekar | 364845265 | Sandesh | Bhat | sabhat@syr.edu | ECS | 3.6 | Cloud |
| Siddhant | Bandiwadekar | 554789654 | Pooja | Parvatkar | pooja@syr.edu | IM | 3.5 | Cloud |
| Siddhant | Bandiwadekar | 569874125 | Bhavya | Gala | bgala@syr.edu | IM | 3.76 | Cloud |

**Project Code**

- **ProjectPython.py**

```
import pandas as pd
from pathlib import Path
import fuzzymatcher

Prof = pd.read_csv (r'/var/www/html/Prof.csv')
print(Prof)


Stud = pd.read_csv (r'/var/www/html/Stud.csv')
#print(Stud)


left_on = ["Department","Course"]
right_on = ["Stud_Department","Stud_Course"]


matched_results = fuzzymatcher.fuzzy_left_join(Prof,
                                               Stud,
                                               left_on,
                                               right_on,
                                               left_id_col='Course',
                                               right_id_col='Stud_Course'
                                                )
```

```
#matched_results


cols = [  "First_Name_left","Last_Name_left", "SU_ID", "First_Name_right", "Last_Name_right",

   "Email_right", "Stud_Department", "GPA", "Stud_Course"]

matched = matched_results[cols].sort_values(by=["First_Name_left"], ascending=True).head(50)


#matched

matched.columns=["Professor First Name","Professor Last Name", "SU ID", "Student First Name",
"Student Last Name",  "Student Email", "Student Department", "GPA", "Course"]

matched.to_csv(r'/var/www/html/login/New.csv', index = False)
```

- Description of code – The above code uses the fuzzymatcher library of python to match the two columns on the basis of a common identifier. In this case "Course" is the common identifier among the two tables. A .csv file with matched records is then exported to the /var/www/html/login/ directory.

- **UpdateProfessor.sh**

```
mysql -h test.cwieffceuygm.us-east-1.rds.amazonaws.com -u admin -D university -p --batch --quick -e
"SELECT * FROM Professor " | sed 's/\t/","/g;s/^/"/;s/$/"/;s/\n//g' > Prof.csv
```

- Description of code – The above code exports the updated Professor Table from the RDS instance and creates a .csv file with separate columns including column names.

- **UpdateStudent.sh**

```
mysql -h test.cwieffceuygm.us-east-1.rds.amazonaws.com -u admin -D university -p --batch --quick -e
"SELECT * FROM Professor " | sed 's/\t/","/g;s/^/"/;s/$/"/;s/\n//g' > Stud.csv
```

- Description of code – The above code exports the updated Student Table from the RDS instance and creates a .csv file with separate columns including column names.

- **Php_insert.php**

```
<?php

$connect = mysqli_connect('test.cwieffceuygm.us-east-1.rds.amazonaws.com', 'admin',
'password','university');

if(!$connect)

{
```

```
echo 'Not Connected To Server';
}
if(!mysqli_select_db($connect,'university'))
{
echo 'Database Not Selected';
}
$First_Name=$_POST['First_Name'];
$Last_Name=$_POST['Last_Name'];
$Emp_ID=$_POST['Emp_ID'];
$Age=$_POST['Age'];
$Email=$_POST['Email'];
$Phone_No=$_POST['Phone_No'];
$Department=$_POST['Department'];
$Course=$_POST['Course'];
$sql="INSERT INTO Professor
(First_Name,Last_Name,Emp_ID,Age,Email,Phone_No,Department,Course)
VALUES('$First_Name','$Last_Name','$Emp_ID','$Age','$Email','$Phone_No','$Department','$Course')";
if(!mysqli_query($connect,$sql))
{
echo 'Value Not Inserted';
}
header("refresh:0;  url=http://52.90.158.69/Success.html");
?>
```

- Description of code – The above code is used to get the records from the Professor Sign Up form and enter the details in the Professor table in our RDS MySQL Server instance.

- **Php_insert1.php**

```
<?php
$connect = mysqli_connect('test.cwieffceuygm.us-east-1.rds.amazonaws.com', 'admin',
'password','university');
if(!$connect)
```

```
{
echo 'Not Connected To Server';
}
if(!mysqli_select_db($connect,'university'))
{
echo 'Database Not Selected';
}
$First_Name=$_POST['First_Name'];
$Last_Name=$_POST['Last_Name'];
$SU_ID=$_POST['SU_ID'];
$Email=$_POST['Email'];
$Stud_Department=$_POST['Stud_Department'];
$GPA=$_POST['GPA'];
$Stud_Course=$_POST['Stud_Course'];
$sql="INSERT INTO Student
(First_Name,Last_Name,SU_ID,Email,Stud_Department,GPA,Stud_Course)
VALUES('$First_Name','$Last_Name','$SU_ID','$Email','$Stud_Department','$GPA','$Stud_Co
urse')";
if(!mysqli_query($connect,$sql))
{
echo 'Value Not Inserted';
}
header("refresh:0; url=http://52.90.158.69/Success1.html");
?>
```

- Description of code – The above code is used to get the records from the Student Sign Up form and enter the details in the Student table in our RDS MySQL Server instance.

- **Log.php**

```
<?php
$connect = mysqli_connect('test.cwieffceuygm.us-east-1.rds.amazonaws.com', 'admin',
'password','university');
if($connect->connect_error)
{
```

```
die('connection failed');
}
$message = "Invalid Username Or Password";
$Email=$_POST['Email'];
$Emp_ID=$_POST['Emp_ID'];
$sql="SELECT Email,Emp_ID FROM Professor  WHERE Email='$Email' AND Emp_ID='$Emp_ID'";
$result=$connect->query($sql);
if($result-> num_rows>0)
{
while($row=$result->fetch_assoc())
{
echo "Admin Name Is " . $row["name"];
 header('Location: http://52.90.158.69/Final.html');
 exit;
}
}
else
echo "<script type='text/javascript'>alert('$message');</script>";
header("Refresh:0; url=http://52.90.158.69/login/index.html");
?>
```

- Description of code – The above code is used to validate and authenticate the login details of the professor which include Email and Employee ID. If the details are correctly validated then the professor gets access to the page that contains the matched records of students.

- **Final.html**

```
<!DOCTYPE html>
<html>
 <head>
  <title>Teaching Assistant Portal</title>
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.0/jquery.min.js"></script>
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css"
/>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
```

```
 </head>
 <body>
  <div class="container">
   <div class="table-responsive">
    <h1 align="center">Teaching Assistant Portal</h1>
    <br />
    <div align="center">
     <button type="button" name="load_data" id="load_data" class="btn btn-info">Show Matching
Records</button>
     <br>
     <br>
     <input type="button" class="btn btn-info" value="Home"
onClick="location.href='http://52.90.158.69/task.html';">
    </div>
    <br />
    <div id="employee_table">
    </div>
   </div>
  </div>
 </body>
</html>

<script>
$(document).ready(function(){
 $('#load_data').click(function(){
  $.ajax({
   url:"http://52.90.158.69/login/New.csv",
   dataType:"text",
   success:function(data)
   {
    var employee_data = data.split(/\r?\n|\r/);
    var table_data = '<table class="table table-bordered table-striped">';
    for(var count = 0; count<employee_data.length-1; count++)
    {
     var cell_data = employee_data[count].split(",");
     table_data += '<tr>';
     for(var cell_count=0; cell_count<cell_data.length; cell_count++)
     {
      if(count === 0)
      {
       table_data += '<th>'+cell_data[cell_count]+'</th>';
      }
      else
      {
       table_data += '<td>'+cell_data[cell_count]+'</td>';
      }
     }
     table_data += '</tr>';
```

```
   }
   table_data += '</table>';
   $('#employee_table').html(table_data);
  }
 });
});

});

</script>
```

- ❖ Description of code – The above code dynamically displays the generated .csv file containing the matched Professor-Student records using jquery ajax method.

- **Project Demo Link**

  https://video.syr.edu/media/t/1_r8ie44ee