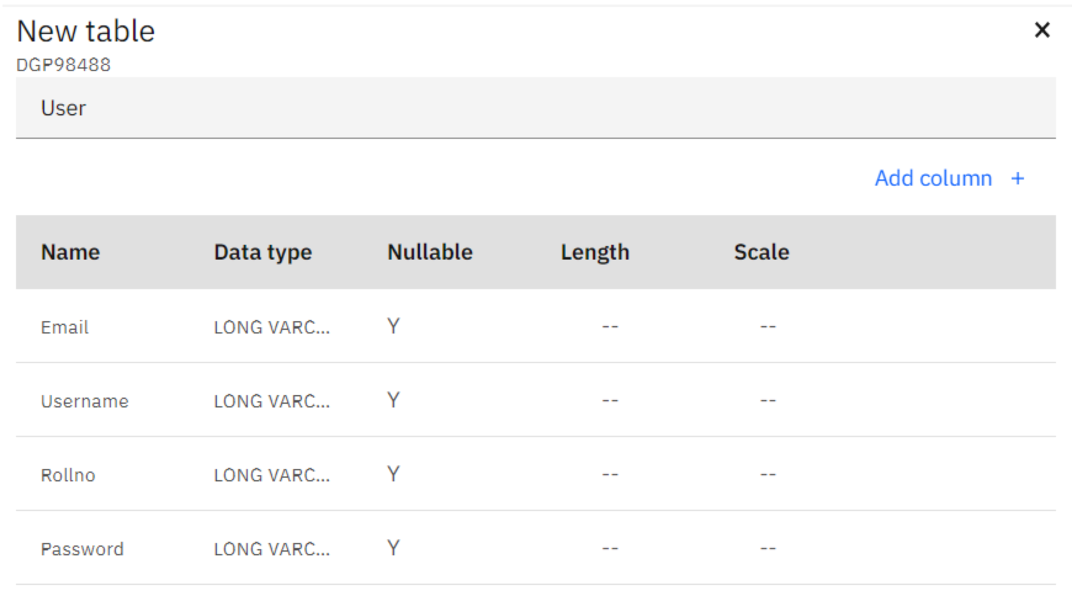
Assignment -2
Flask program connected to db2

Assignment Date	19 September 2022
Student Name	N. Ramya
Student Roll Number	2019115076
Maximum Marks	2 Marks

Q1. Create User table with user with email, username, roll number, password.

Solution:

Create table user (Email long varchar, username long varchar, rollno long varchar, password long varchar);

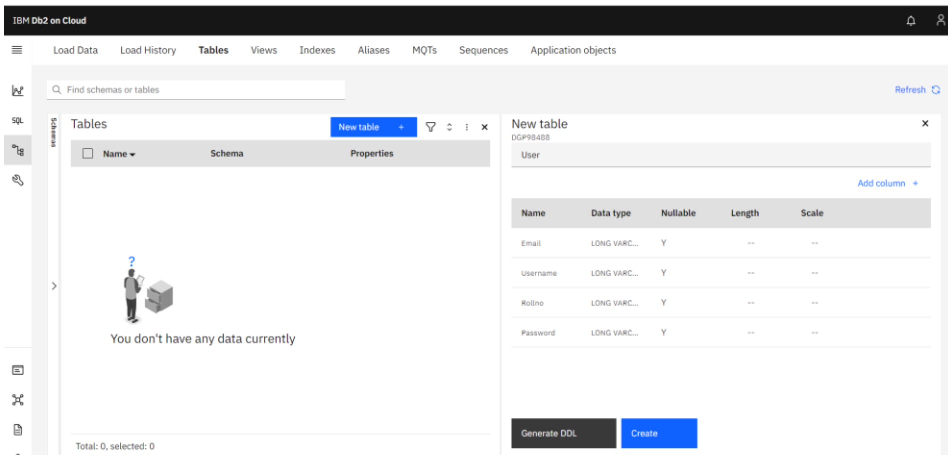

New table

DGP98488

User

Add column +

Name	Data type	Nullable	Length	Scale
Email	LONG VARC...	Y	--	--
Username	LONG VARC...	Y	--	--
Rollno	LONG VARC...	Y	--	--
Password	LONG VARC...	Y	--	--



You don't have any data currently

Total: 0, selected: 0

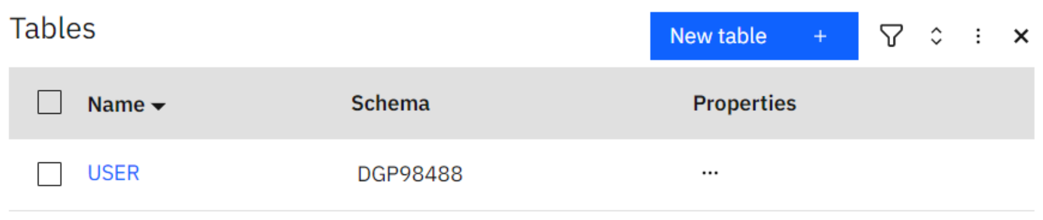
New table

DGP98488

User

Add column +

Name	Data type	Nullable	Length	Scale
Email	LONG VARC...	Y	--	--
Username	LONG VARC...	Y	--	--
Rollno	LONG VARC...	Y	--	--
Password	LONG VARC...	Y	--	--



Name	Schema	Properties
USER	DGP98488	···

Q2. Perform UPDATE, DELETE Queries with user table.

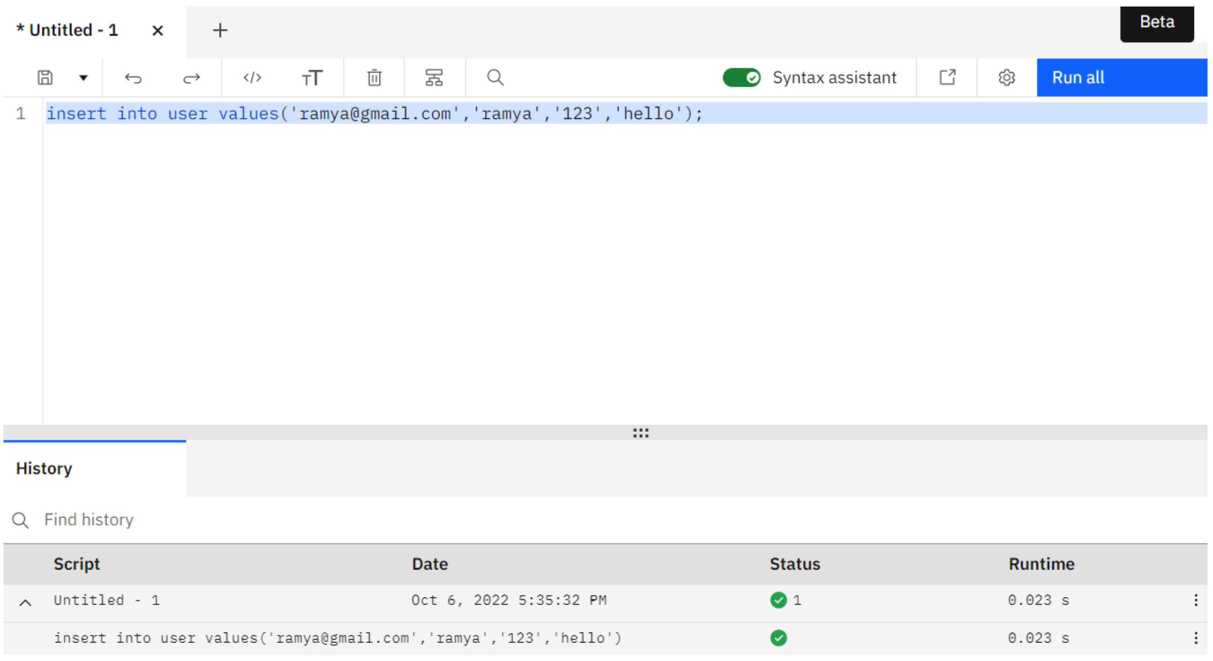
Solution:

Insert into user values('ramya@gmail.com','ramya','123','hello');

Update user set email='ramya123@gmail.com' where username='ramya';

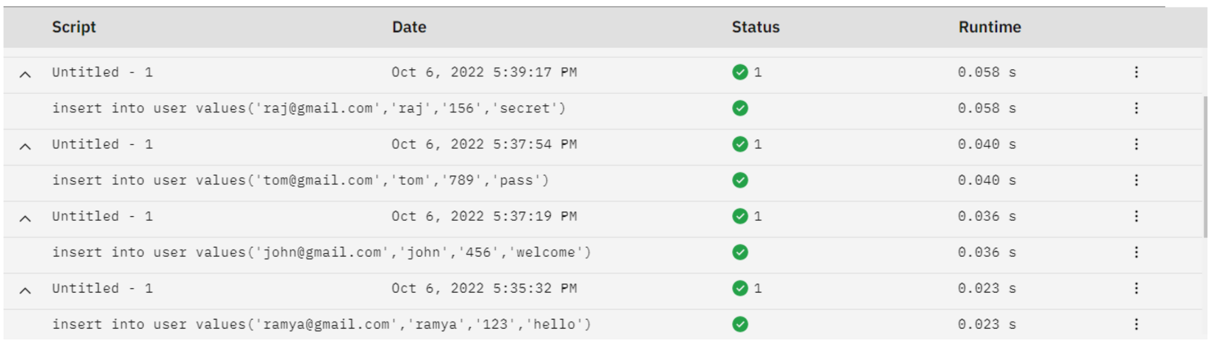
Update user set password='hello123' where username = 'john';

Delete from user where username='teena';

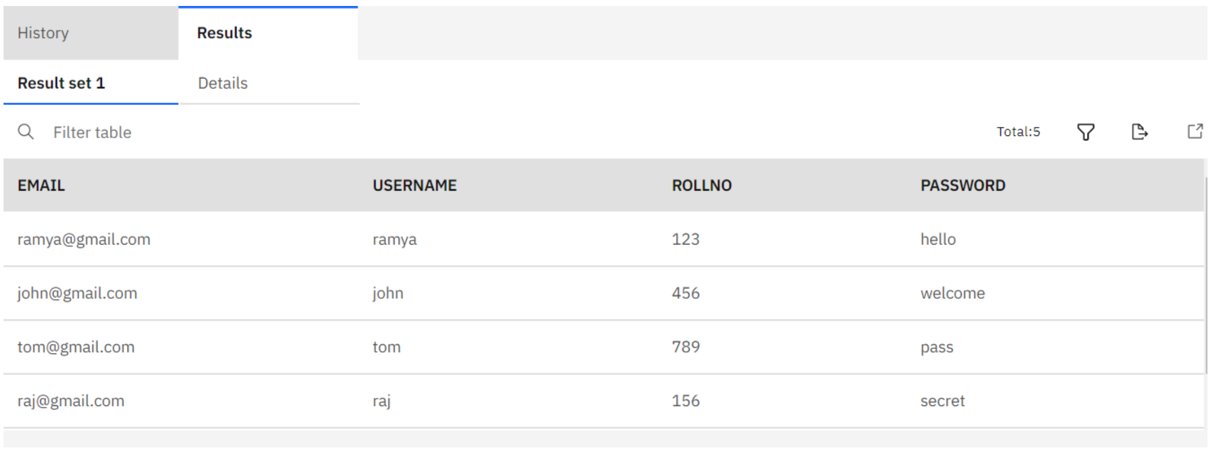


```
insert into user values('ramya@gmail.com','ramya','123','hello');
```

Script	Date	Status	Runtime
Untitled - 1	Oct 6, 2022 5:35:32 PM	✓ 1	0.023 s
insert into user values('ramya@gmail.com','ramya','123','hello')		✓	0.023 s



Script	Date	Status	Runtime
Untitled - 1	Oct 6, 2022 5:39:17 PM	✓ 1	0.058 s
insert into user values('raj@gmail.com','raj','156','secret')		✓	0.058 s
Untitled - 1	Oct 6, 2022 5:37:54 PM	✓ 1	0.040 s
insert into user values('tom@gmail.com','tom','789','pass')		✓	0.040 s
Untitled - 1	Oct 6, 2022 5:37:19 PM	✓ 1	0.036 s
insert into user values('john@gmail.com','john','456','welcome')		✓	0.036 s
Untitled - 1	Oct 6, 2022 5:35:32 PM	✓ 1	0.023 s
insert into user values('ramya@gmail.com','ramya','123','hello')		✓	0.023 s

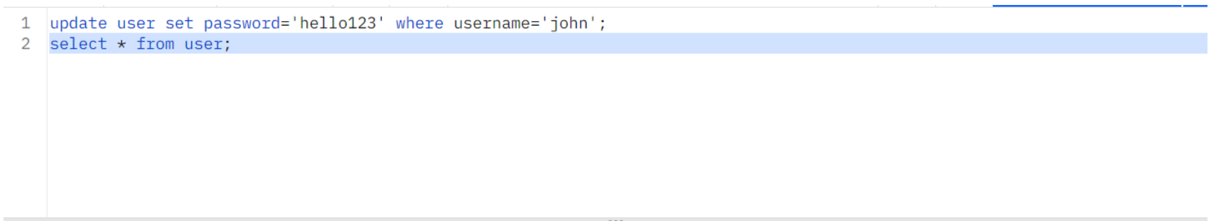


History | Results

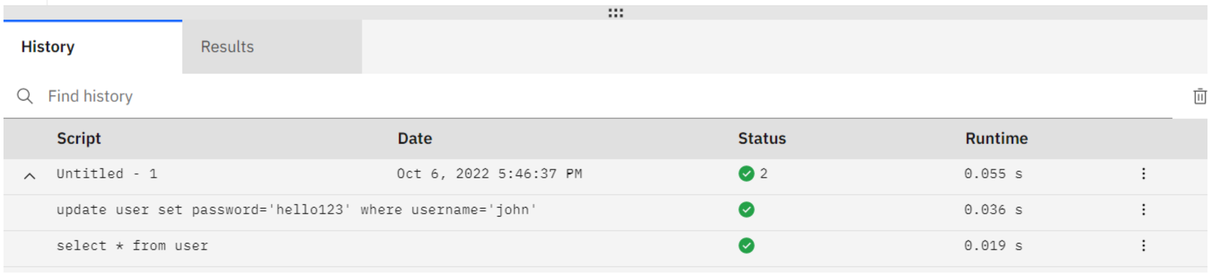
Result set 1 | Details

Filter table — Total:5

EMAIL	USERNAME	ROLLNO	PASSWORD
ramya@gmail.com	ramya	123	hello
john@gmail.com	john	456	welcome
tom@gmail.com	tom	789	pass
raj@gmail.com	raj	156	secret



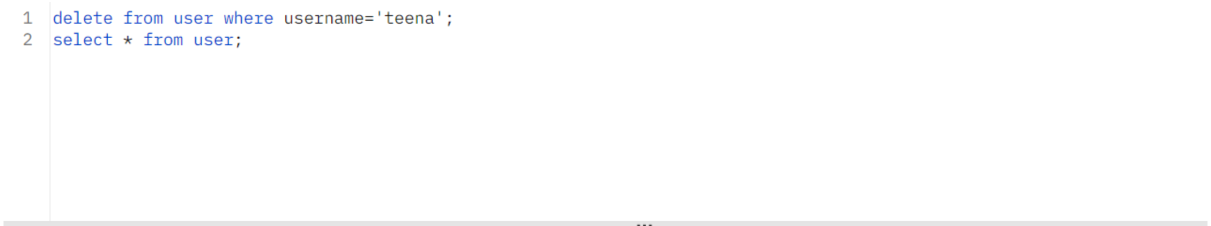
```sql
update user set password='hello123' where username='john';
select * from user;
```



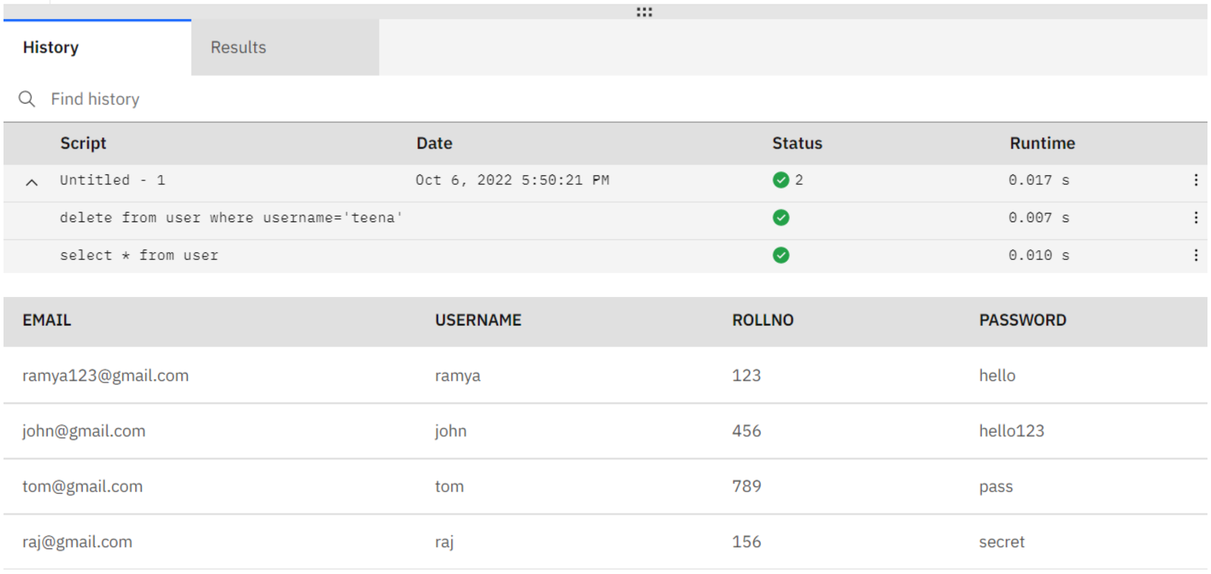
History | Results

Find history

Script	Date	Status	Runtime
Untitled - 1	Oct 6, 2022 5:46:37 PM	✓ 2	0.055 s
update user set password='hello123' where username='john'		✓	0.036 s
select * from user		✓	0.019 s



```sql
delete from user where username='teena';
select * from user;
```



History | Results

Find history

Script	Date	Status	Runtime
Untitled - 1	Oct 6, 2022 5:50:21 PM	✓ 2	0.017 s
delete from user where username='teena'		✓	0.007 s
select * from user		✓	0.010 s

EMAIL	USERNAME	ROLLNO	PASSWORD
ramya123@gmail.com	ramya	123	hello
john@gmail.com	john	456	hello123
tom@gmail.com	tom	789	pass
raj@gmail.com	raj	156	secret

Q3. Connect python code to db2.

Solution:

```
from flask import Flask,render_template,request,redirect,url_for,session

import ibm_db

import re


app=Flask(__name__)

app.secret_key='a'


conn=ibm_db. connect("DATABASE=bludb;HOSTNAME=b70af05b-76e4-4bca-a1f5-23dbb4c6a74e.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32716;Security=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=dgp98488;PWD=T4ZhPsFfVgztgO7H;", "","")
```

Q4. Create a flask app with registration page, login page and welcome page. By default, load the registration page once the user enters all the fields store the data in database and navigate to login page authenticate user username and password. If the user is valid show the welcome page.

Solution:

```
from flask import Flask,render_template,request,redirect,url_for,session

import ibm_db

import re


app=Flask(__name__)

app.secret_key='a'


conn=ibm_db.connect("DATABASE=bludb;HOSTNAME=b70af05b-76e4-4bca-a1f5-23dbb4c6a74e.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32716;Security=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=dgp98488;PWD=T4ZhPsFfVgztgO7H;", "","")


@app.route('/')

def home():

    return render_template('register.html')
```

```
@app.route("/login",methods=['GET','POST'])
def login():
    global userid
    msg=" "

    if request.method=="POST":
        username=request.form['username']
        password=request.form['password']
        sql="SELECT * from user where username=? AND password=?"
        stmt=ibm_db.prepare(conn,sql)
        ibm_db.bind_param(stmt,1,username)
        ibm_db.bind_param(stmt,2,password)
        ibm_db.execute(stmt)
        account=ibm_db.fetch_assoc(stmt)
        print(account)
        if account:
            session['Loggedin']=True
            session['id']=account['USERNAME']
            userid=account['USERNAME']
            session['username']=account['USERNAME']
            msg='Logged in successfully!!'
            return
render_template("dashboard.html",username=request.form['username'],msg=msg)
        else:
            msg="Incorrect Username/Password"
    return render_template('login.html',msg=msg)

@app.route("/register",methods=["GET","POST"])
def register():
    msg=" "
    if request.method=="POST":
        username=request.form['username']
        email=request.form['email']
        rollno=request.form['rollno']
        password=request.form['password']
        sql="SELECT * FROM user where username=?"
        stmt=ibm_db.prepare(conn,sql)
        ibm_db.bind_param(stmt,1,username)
        ibm_db.execute(stmt)
        account=ibm_db.fetch_assoc(stmt)
        print(account)
        if account:
            msg="Account already exists!!"

        elif not re.match(r'[^@]+@[^@]+\.[^@]+',email):
            msg="Invalid Email Address!"
        elif not re.match(r'([a-zA-Z]|[0-9])+',username):
            msg="Name must contain only characters and numbers!"
        else:
            insert_sql="INSERT into user values(?,?,?,?)"
            prep_stmt=ibm_db.prepare(conn,insert_sql)
            ibm_db.bind_param(prep_stmt,1,email)
            ibm_db.bind_param(prep_stmt,2,username)
            ibm_db.bind_param(prep_stmt,3,rollno)
            ibm_db.bind_param(prep_stmt,4,password)
            ibm_db.execute(prep_stmt)
            msg="You have registered successfully"
            return render_template('login.html',msg=msg)

        return render_template('register.html',msg=msg)

@app.route('/dashboard')
```

def dash():

    return render_template('dashboard.html')

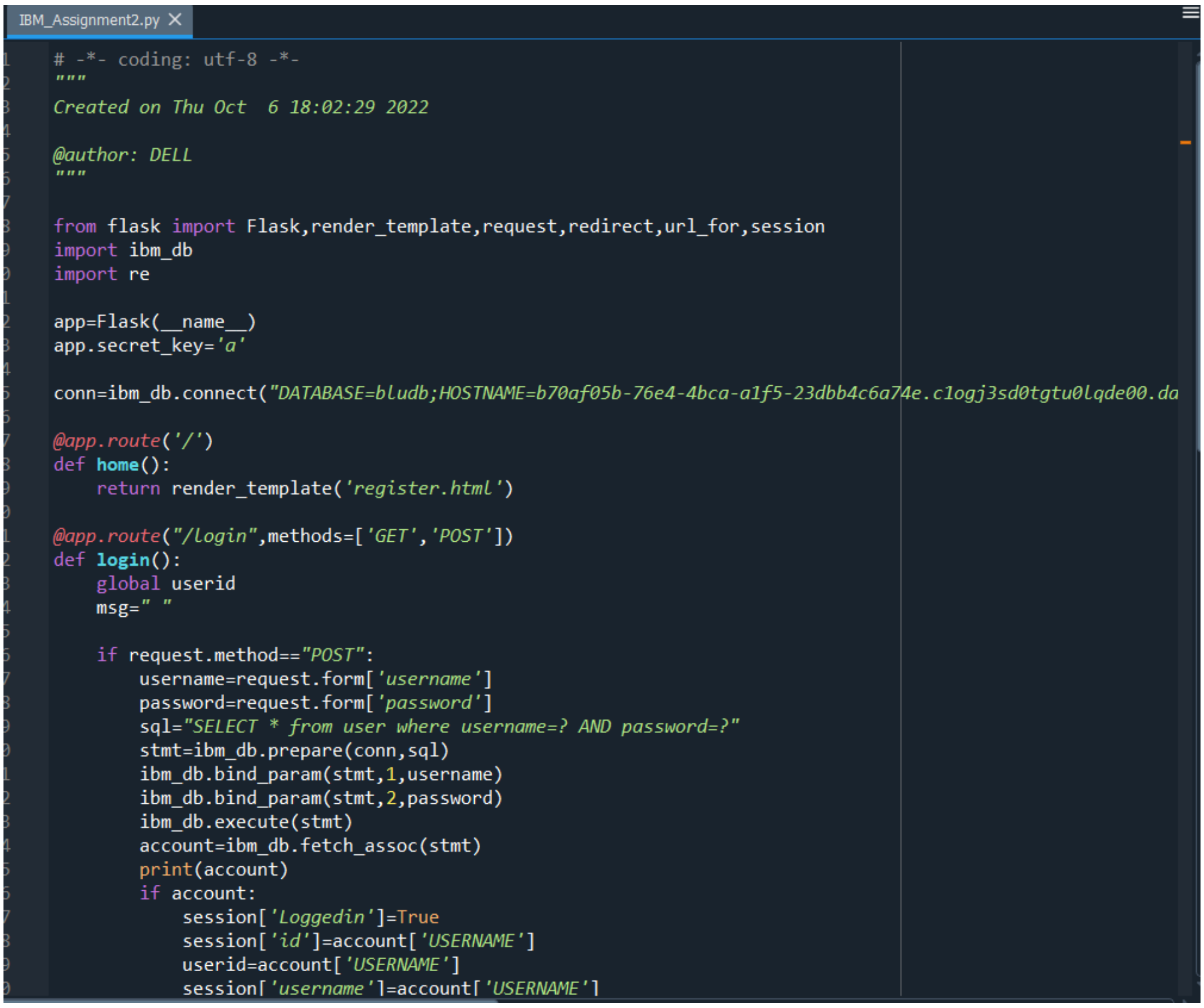
@app.route('/display')

def display():

    print (session["username"],session["id"])

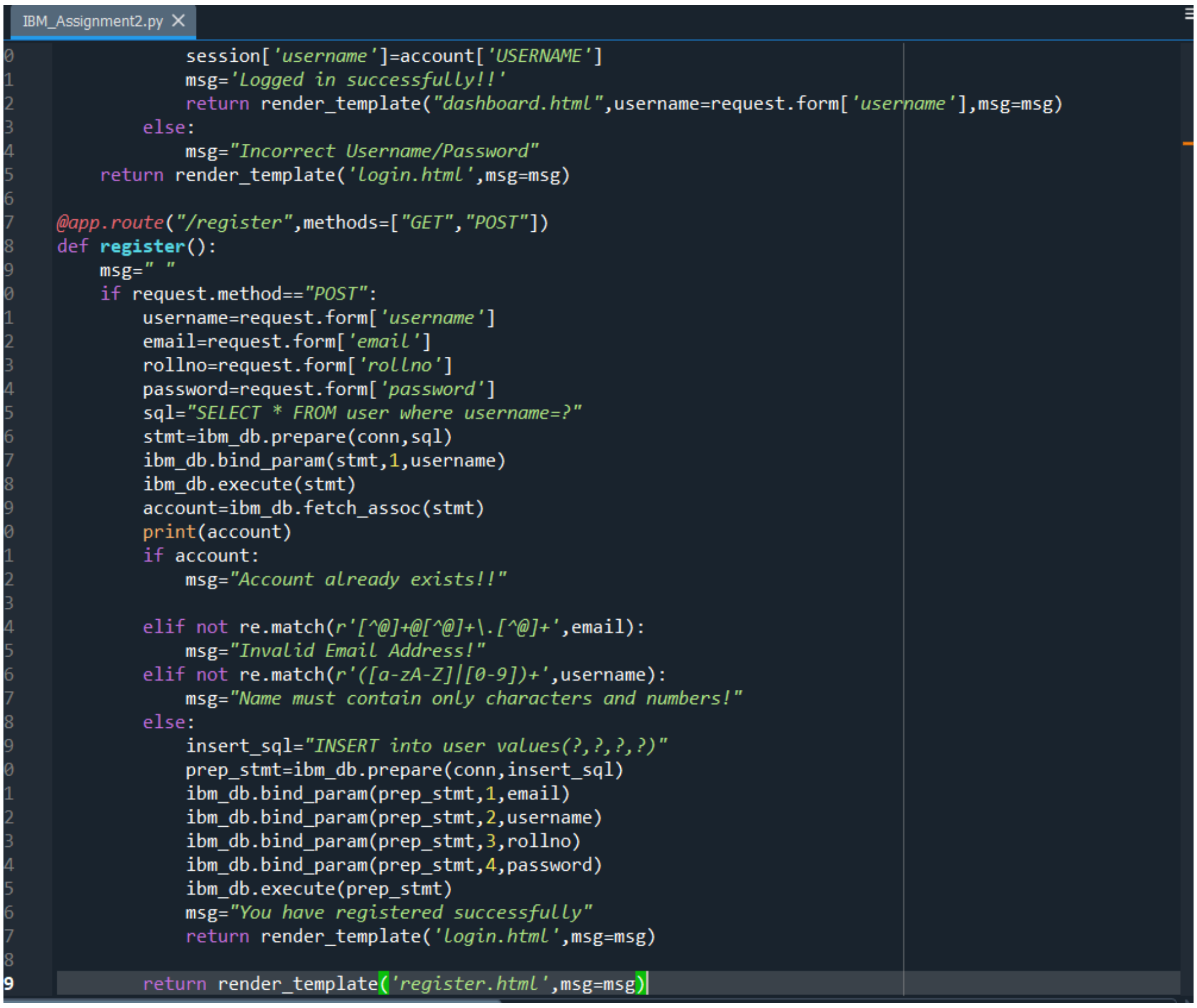
if __name__ == "__main__":

    app.run(debug=True)

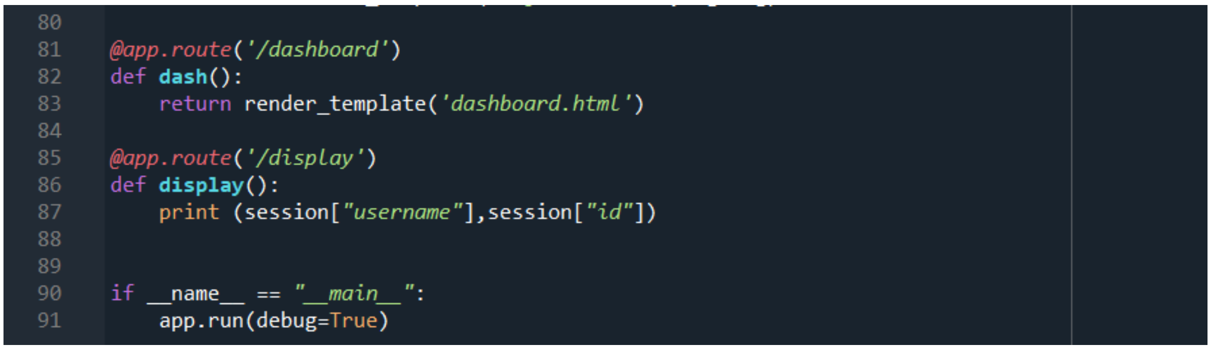


IBM_Assignment2.py

```
# -*- coding: utf-8 -*-
"""
Created on Thu Oct  6 18:02:29 2022

@author: DELL
"""

from flask import Flask,render_template,request,redirect,url_for,session
import ibm_db
import re

app=Flask(__name__)
app.secret_key='a'

conn=ibm_db.connect("DATABASE=bludb;HOSTNAME=b70af05b-76e4-4bca-a1f5-23dbb4c6a74e.c1ogj3sd0tgtu0lqde00.da

@app.route('/')
def home():
    return render_template('register.html')

@app.route("/login",methods=['GET','POST'])
def login():
    global userid
    msg=" "

    if request.method=="POST":
        username=request.form['username']
        password=request.form['password']
        sql="SELECT * from user where username=? AND password=?"
        stmt=ibm_db.prepare(conn,sql)
        ibm_db.bind_param(stmt,1,username)
        ibm_db.bind_param(stmt,2,password)
        ibm_db.execute(stmt)
        account=ibm_db.fetch_assoc(stmt)
        print(account)
        if account:
            session['Loggedin']=True
            session['id']=account['USERNAME']
            userid=account['USERNAME']
            session['username']=account['USERNAME']
```

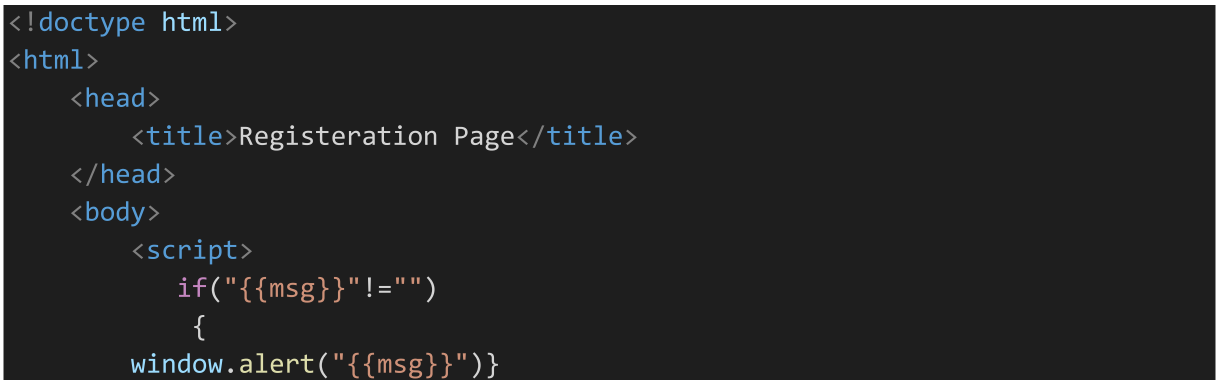


```
            session['username']=account['USERNAME']
            msg='Logged in successfully!!'
            return render_template("dashboard.html",username=request.form['username'],msg=msg)
        else:
            msg="Incorrect Username/Password"
    return render_template('login.html',msg=msg)

@app.route("/register",methods=["GET","POST"])
def register():
    msg=" "
    if request.method=="POST":
        username=request.form['username']
        email=request.form['email']
        rollno=request.form['rollno']
        password=request.form['password']
        sql="SELECT * FROM user where username=?"
        stmt=ibm_db.prepare(conn,sql)
        ibm_db.bind_param(stmt,1,username)
        ibm_db.execute(stmt)
        account=ibm_db.fetch_assoc(stmt)
        print(account)
        if account:
            msg="Account already exists!!"

        elif not re.match(r'[^@]+@[^@]+\.[^@]+',email):
            msg="Invalid Email Address!"
        elif not re.match(r'([a-zA-Z]|[0-9])+',username):
            msg="Name must contain only characters and numbers!"
        else:
            insert_sql="INSERT into user values(?,?,?,?)"
            prep_stmt=ibm_db.prepare(conn,insert_sql)
            ibm_db.bind_param(prep_stmt,1,email)
            ibm_db.bind_param(prep_stmt,2,username)
            ibm_db.bind_param(prep_stmt,3,rollno)
            ibm_db.bind_param(prep_stmt,4,password)
            ibm_db.execute(prep_stmt)
            msg="You have registered successfully"
            return render_template('login.html',msg=msg)

        return render_template('register.html',msg=msg)
```



```

@app.route('/dashboard')
def dash():
    return render_template('dashboard.html')

@app.route('/display')
def display():
    print (session["username"],session["id"])


if __name__ == "__main__":
    app.run(debug=True)
```

Register.html



```
<!doctype html>
<html>
    <head>
        <title>Registeration Page</title>
    </head>
    <body>
        <script>
           if("{{msg}}"!="")
            {
        window.alert("{{msg}}")}
```

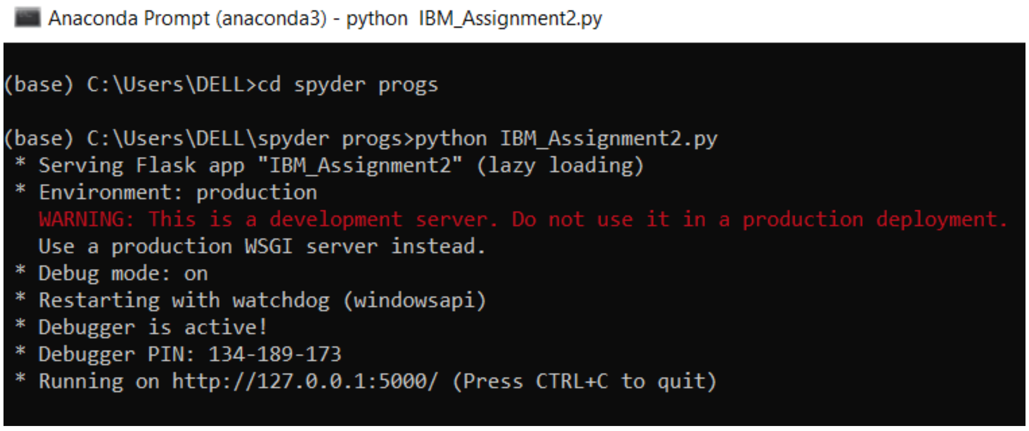
```
            </script>
        <form action="/register" method="POST">
            <div>
                <div>
                    <label for="username">Username: </label>
                    <input type="text" name="username" id="username" />

                </div>
                <div>
                    <label for="rollno">Roll Number: </label>
                    <input type="number" name="rollno" id="rollno" />
                </div>
                <div>
                    <label for="email">Email: </label>
                    <input type="email" name="email" id="email" />
                </div>
                <div>
                    <label for="password">Password: </label>
                    <input type="password" name="password" id="password" />
                </div>
                <div>
                    <input type="submit" value="Submit" />
                </div>
            </div>
        </form>
    </body>
</html>
```

Login.html

```
<!doctype html>
<html>
    <head>
        <title>Login Page</title>
    </head>
    <body>
        <script>
            if("{{msg}}"!="")
            {
        window.alert("{{msg}}")}
        </script>
        <form action="/login" method="POST">
            <div>
                <div>
                    <label for="username">Username: </label>
                    <input type="text" name="username" id="username" />

                </div>
                <div>
                    <label for="password">Password: </label>
                    <input type="password" name="password" id="password" />
                </div>
                <div>
                    <input type="submit" value="Submit" />
                </div>
            </div>
        </form>
    </body>
</html>
```

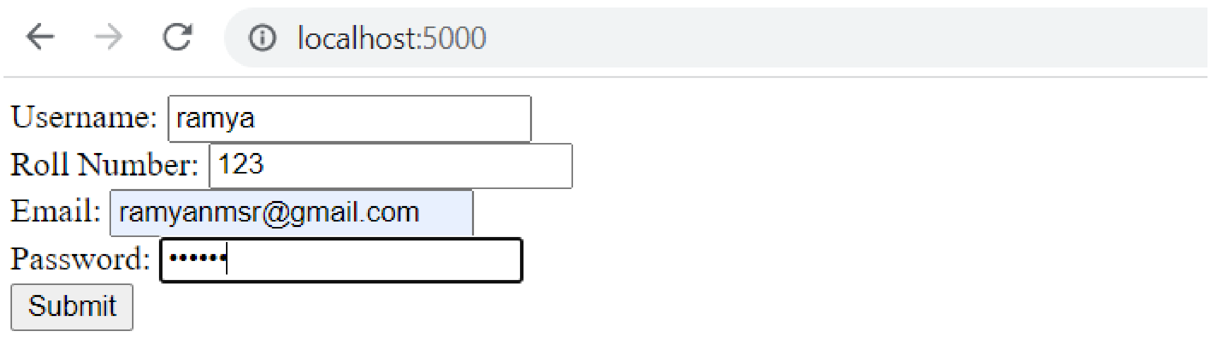
Dashboard.html

```
<!doctype html>
<html>
    <head>
        <title>Dashboard</title>
    </head>
    <body>
        <script>
            if("{{msg}}"!="")
            {
        window.alert("{{msg}}")}
            </script>
       Welcome to the dashboard {{username}} !!
    </body>
</html>
```

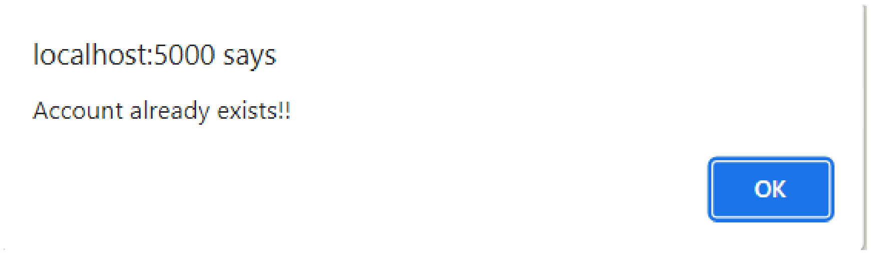


Anaconda Prompt (anaconda3) - python IBM_Assignment2.py

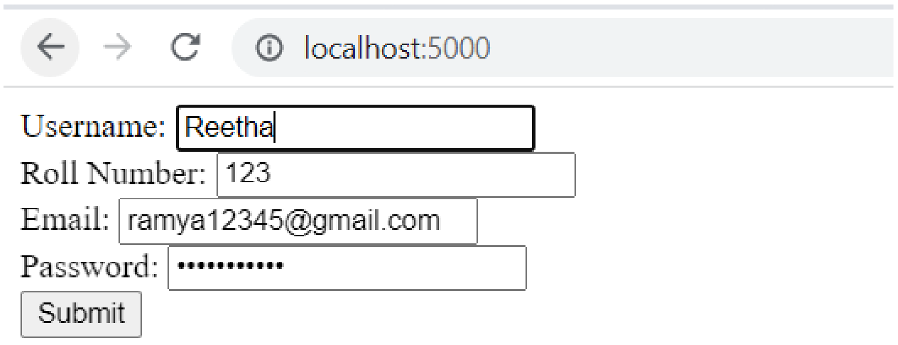
```
(base) C:\Users\DELL>cd spyder progs

(base) C:\Users\DELL\spyder progs>python IBM_Assignment2.py
 * Serving Flask app "IBM_Assignment2" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with watchdog (windowsapi)
 * Debugger is active!
 * Debugger PIN: 134-189-173
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```


localhost:5000

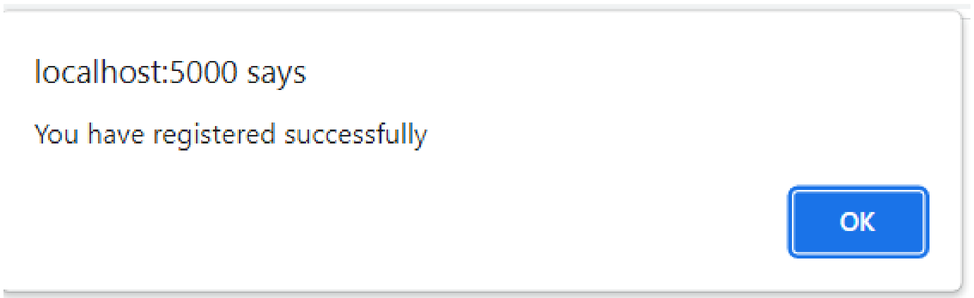
Username: ramya
Roll Number: 123
Email: ramyanmsr@gmail.com
Password: ••••••


localhost:5000 says

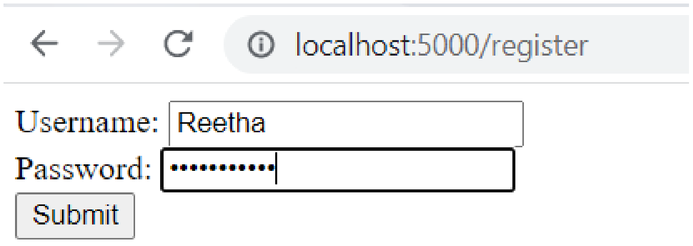
Account already exists!!


localhost:5000

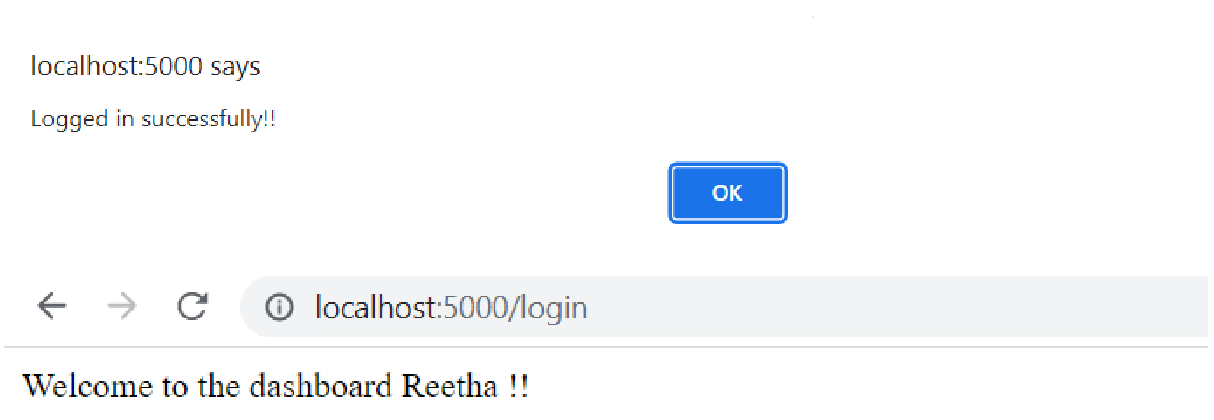
Username: Reetha
Roll Number: 123
Email: ramya12345@gmail.com
Password: •••••••••••


localhost:5000 says

You have registered successfully


localhost:5000/register

Username: Reetha
Password: •••••••••••


```
localhost:5000 says
Logged in successfully!!
```

```
Welcome to the dashboard Reetha !!
```