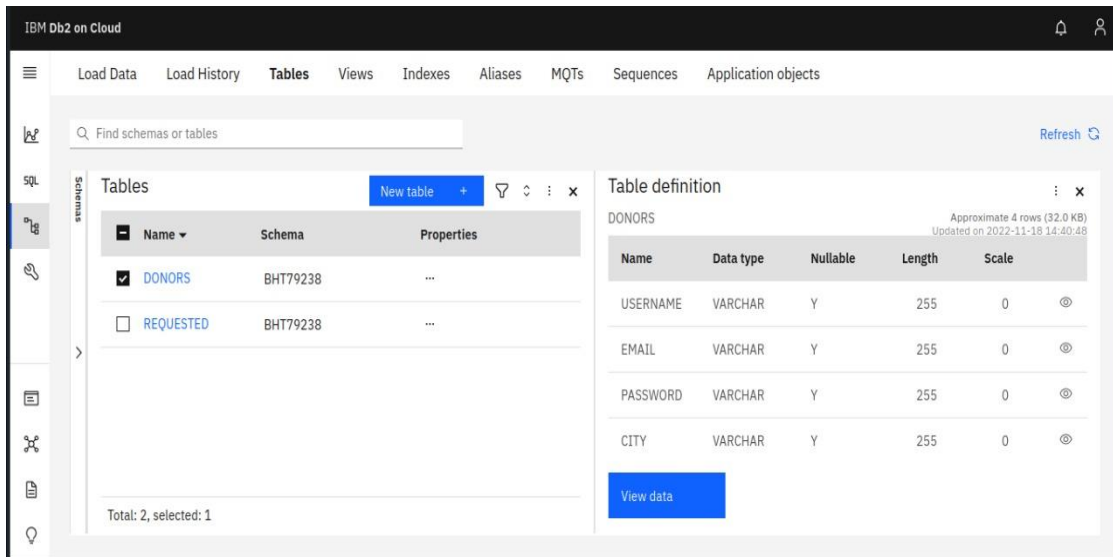
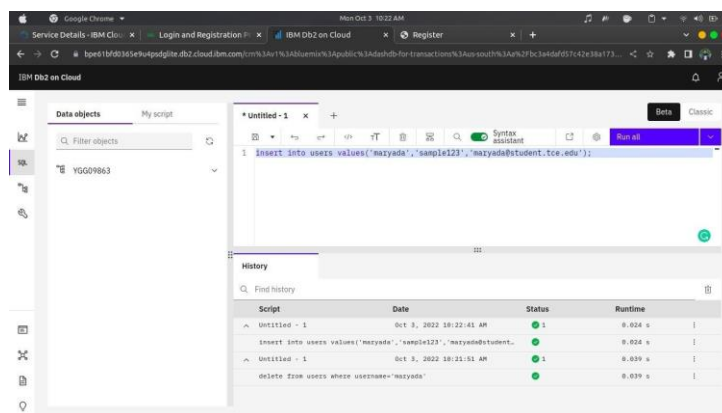


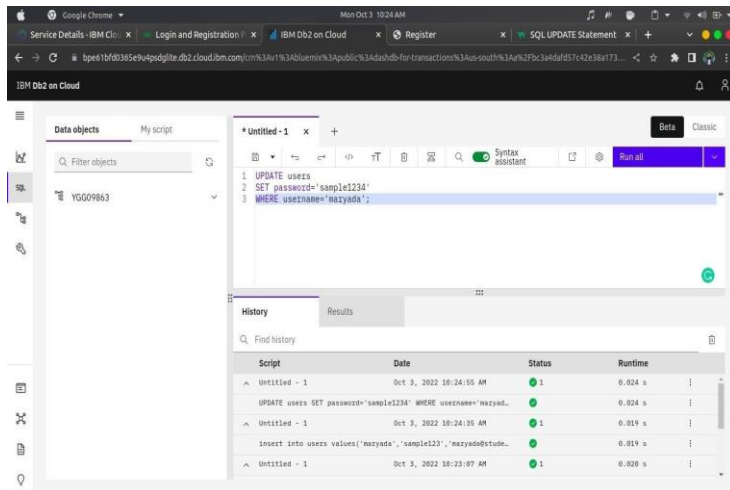
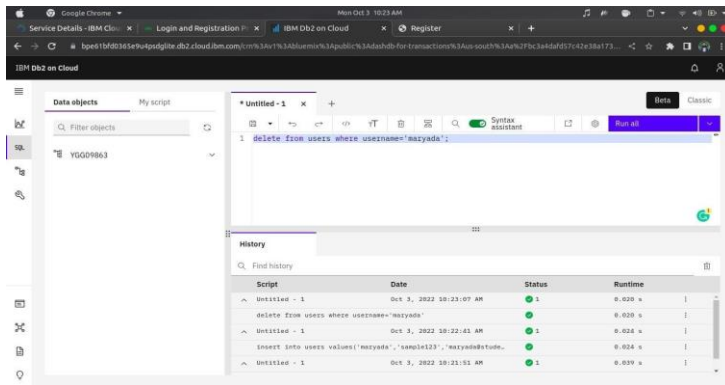
Assignment 2:
917719C024
917719C073
917719C095
917719C112

1. Create User table with user with email,usernameand password.



2. Perform UPDATE,DELETE Queries with user table





The screenshot shows the IBM Db2 on Cloud console interface. The 'Tables' tab is selected, and the table 'BHT79238.DONORS' is displayed. The table has the following columns: USERNAME, EMAIL, PASSWORD, CITY, INFECT, BLOOD, and PHONE. The table contains 5 rows of data. A 'Back' button is visible in the top right corner. An 'Export to CSV' button is visible in the top right corner.

USERNAME	EMAIL	PASSWORD	CITY	INFECT	BLOOD	PHONE
Gokul	gokulap.tce@gmail.com	GokulPass67@	Madurai	uninfected	B Positive	8056514097
Shivanesh	shivanesh@student.tce.edu	Qwerty897	Nilakotai	uninfected	AB Positive	9856478354
Varun	varun@gmail.com	V@rUn563!	Jammu	uninfected	A Negative	7650643756
prithivi	prithiviraj@student.tce.edu	Prithiviraj123	Madurai	uninfected	A Positive	8905612876

3. Connect python code to db2. (Done in question 4)

4. 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

app.py:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
```

""""

Created on Sun Oct 2 12:32:54 2022

@author:prithivi""

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

```
app=Flask(__name__)
app.secret_key='a'
conn=ibm_db.connect("DATABASE=bludb;HOSTNAME=9938aec0-8105-433e-8bf9-
0fbb7e483086.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32459;Security=SSL;SSL
ServerCertificate=DigiCertGlobalRootCA.crt;UID=ygg09863;PWD=dQXLEcjtaCqXsJUt;","","")
```

```
@app.route('/')
@app.route('/login',methods=['GET','POST'])
```

```
def login():
```

```
    global userid;
    msg=" "
```

```
    if request.method == 'POST' and 'username' in request.form and 'password' in request.form:
```

```
        username=request.form['username']
        password=request.form['password']
        sql="SELECT * FROM users 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("index.html", msg=msg)
```

```
    else :
        msg="Incorrect username and password"
```

```
    return render_template("login.html", msg=msg)
```

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

```
def logout():
```

```
    session.pop('loggedin', None)
    session.pop('id', None)
    session.pop('username', None)
    return redirect(url_for('login'))
```

```
@app.route("/register",methods=["GET","POST"])
```

```

def register():
    msg=""
    if request.method == 'POST' and 'username' in request.form and 'password' in request.form and
'email' in request.form :
        username=request.form['username']
        email=request.form['email']
        password=request.form['password']
        sql="SELECT * from users 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-z0-9]+', username):
            msg = 'Username must contain only characters and numbers !'
        elif not username or not password or not email:
            msg = 'Please fill out the form !'
        else:
            insert_sql="INSERT INTO users VALUES(?,?,?)"
            prep_stmt=ibm_db.prepare(conn,insert_sql)
            ibm_db.bind_param(prepare_stmt,1,username)
            ibm_db.bind_param(prepare_stmt,2,email)
            ibm_db.bind_param(prepare_stmt,3,password)
            ibm_db.execute(prepare_stmt)
            msg="You have successfully logged in!"

    elif request.method=="POST":
        msg="Please fill out the form"
    return render_template('register.html', msg=msg)

if __name__=="_main_":
    app.run(host='0.0.0.0')

```

templates folder:

index.html

<!-- Store this code in 'index.html' file inside the 'templates' folder-->

```

<html>
    <head>
        <meta charset="UTF-8">
        <title> Index </title>
        <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">

    </head>
    <body><br></br></br></br></br>
        <div align="center">
            <div align="center" class="border">

```

```

        <div class="header">
            <h1 class="word">Index</h1>
        </div></br></br></br>
        <h1 class="bottom">
            Hi {{ session.username }}!!</br></br> Welcome to the index
page...

        </h1></br></br></br>
        <a href="{{ url_for('logout') }}" class="btn">Logout</a>

    </div>
</div>
</body>
</html>

```

login.html:

<!-- Store this code in 'login.html' file inside the 'templates' folder -->

```

<html>
    <head>
        <meta charset="UTF-8">
        <title> Login </title>
        <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
    </head>
    <body></br></br></br></br></br>
        <div align="center">
            <div align="center" class="border">
                <div class="header">
                    <h1 class="word">Login</h1>
                </div></br></br></br>
                <h2 class="word">
                    <form action="{{ url_for('login') }}" method="post">
                        <div class="msg">{{ msg }}</div>
                        <input id="username" name="username" type="text"
placeholder="Enter Your Username" class="textbox"/></br></br>
                        <input id="password" name="password" type="password"
placeholder="Enter Your Password" class="textbox"/></br></br></br>
                        <input type="submit" class="btn" value="Sign In"></br></br>
                    </form>
                </h2>
                <p class="bottom">Don't have an account? <a class="bottom"
href="{{ url_for('register') }}"> Sign Up here</a></p>
            </div>
        </div>
    </body>
</html>

```

register.html:

<!-- Store this code in 'register.html' file inside the 'templates' folder -->

```

<html>
    <head>
        <meta charset="UTF-8">

```

```

        <title> Register </title>
        <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
</head>
<body><br></br></br></br></br>
    <div align="center">
        <div align="center" class="border">
            <div class="header">
                <h1 class="word">Register</h1>
            </div><br></br></br>
            <h2 class="word">
                <form action="{{ url_for('register') }}" method="post">
                    <div class="msg">{{ msg }}</div>
                        <input id="username" name="username" type="text"
placeholder="Enter Your Username" class="textbox"/></br></br>
                        <input id="password" name="password" type="password"
placeholder="Enter Your Password" class="textbox"/></br></br>
                        <input id="email" name="email" type="text"
placeholder="Enter Your Email ID" class="textbox"/></br></br>
                        <input type="submit" class="btn" value="Sign Up"></br>
                    </form>
                </h2>
                <p class="bottom">Already have an account? <a class="bottom"
href="{{ url_for('login') }}"> Sign In here</a></p>
            </div>
        </div>
    </body>
</html>

```

static folder:
style.css

/* Store this code in 'style.css' file inside the 'static' folder*/

```

.header{
    padding: 5px 120px;
    width: 150px;
    height: 70px;
    background-color: #236B8E;
}

.border{
    padding: 80px 50px;
    width: 400px;
    height: 450px;
    border: 1px solid #236B8E;
    border-radius: 0px;
    background-color: #9AC0CD;
}

.btn {
    padding: 10px 40px;
    background-color: #236B8E;
}

```

```
        color: #FFFFFFF;
        font-style: oblique;
        font-weight: bold;
        border-radius: 10px;
    }

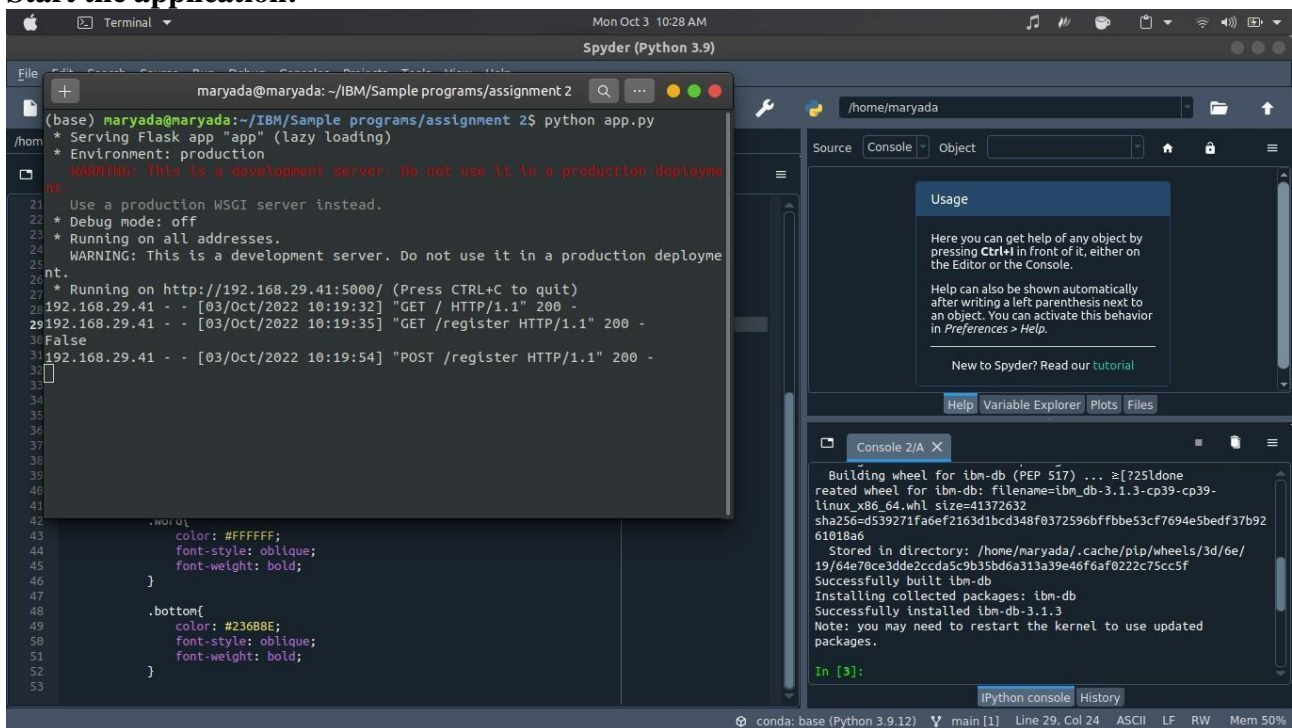
    .textbox{
        padding: 10px 40px;
        background-color: #236B8E;
        text-color: #FFFFFFF;
        border-radius: 10px;
    }

    ::placeholder {
        color: #FFFFFFF;
        opacity: 1;
        font-style: oblique;
        font-weight: bold;
    }

    .word{
        color: #FFFFFFF;
        font-style: oblique;
        font-weight: bold;
    }

    .bottom{
        color: #236B8E;
        font-style: oblique;
        font-weight: bold;
    }
```


Output: Start the application:



The screenshot shows the Spyder Python IDE interface. The top-left pane displays a terminal window with the following output:

```
(base) maryada@maryada: ~/IBM/Sample programs/assignment 2$ python app.py
* Serving Flask app "app" (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: off
* Running on all addresses.
WARNING: This is a development server. Do not use it in a production deployment.
* Running on http://192.168.29.41:5000/ (Press CTRL+C to quit)
192.168.29.41 - - [03/Oct/2022 10:19:32] "GET / HTTP/1.1" 200 -
192.168.29.41 - - [03/Oct/2022 10:19:35] "GET /register HTTP/1.1" 200 -
False
192.168.29.41 - - [03/Oct/2022 10:19:54] "POST /register HTTP/1.1" 200 -
```

The bottom-left pane shows a code editor with the following CSS code:

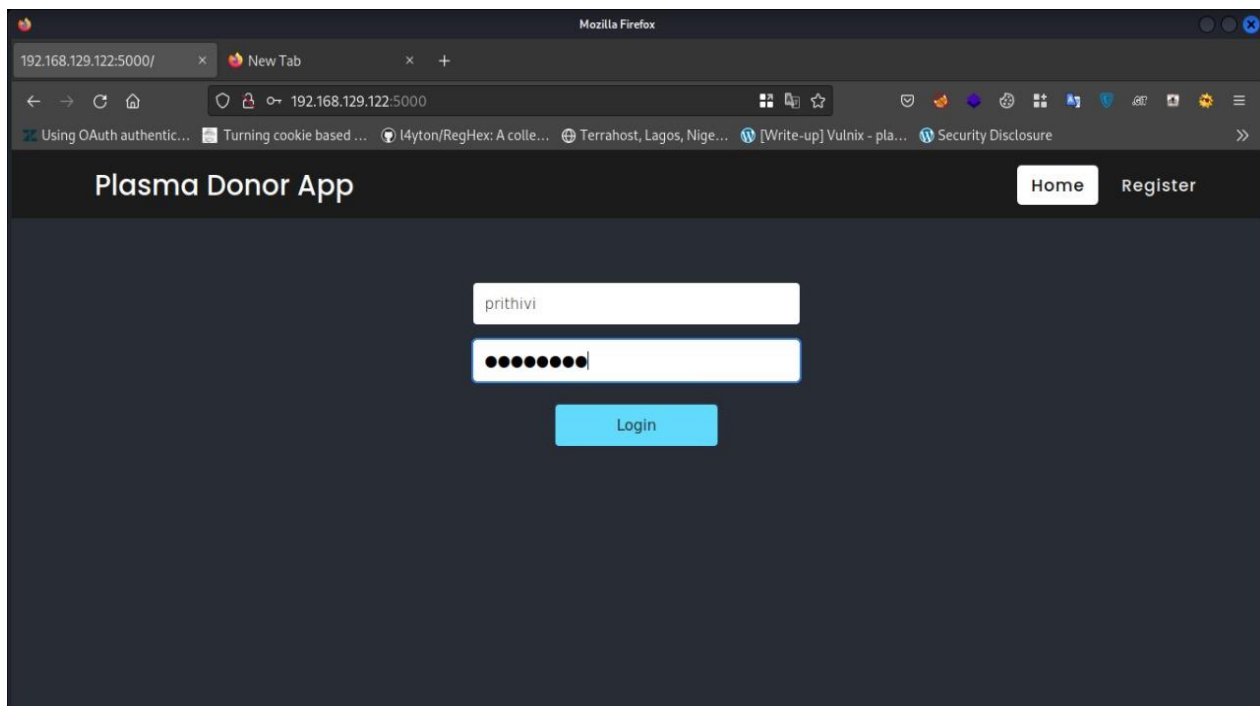
```
.top{
    color: #FFFFFF;
    font-style: oblique;
    font-weight: bold;
}

.bottom{
    color: #236B8E;
    font-style: oblique;
    font-weight: bold;
}
```

The right-hand pane shows the IPython console with the following output:

```
Building wheel for ibm-db (PEP 517) ... [?25ldone
reated wheel for ibm-db: filename=ibm_db-3.1.3-cp39-cp39-
linux_x86_64.whl size=41372632
sha256=d539271fa9ef2163d1bcd348f0372596bffb53cf7694e5bedf37b92
61018a6
Stored in directory: /home/maryada/.cache/pip/wheels/3d/6e/
19/64e70ce3dde2ccda5c9b35bd6a313a39e46fa0222c75cc5f
Successfully built ibm-db
Installing collected packages: ibm-db
Successfully installed ibm-db-3.1.3
Note: you may need to restart the kernel to use updated
packages.
In [3]:
```

We are invited with login page.



Registration:

Mozilla Firefox

192.168.129.122:5000/registr... x New Tab x +

192.168.129.122:5000/registration 90%

Using OAuth authentic... Turning cookie based ... l4yton/RegHex: A colle... Terrahost, Lagos, Nige... [Write-up] Vulnix - pla... Security Disclosure

Plasma Donor App

Home

Shivanesh

shivan@telegmail.com

8056514097

Madurai

Uninfected

B Positive

●●●●●●●●

Register

IBM Db2 on Cloud

Load Data Load History **Tables** Views Indexes Aliases MQTs Sequences Application objects

BHT79238.REQUESTED Back

Export to CSV

BLOODGRP	ADDRESS	NAME	EMAIL	PHONE
A Positive	170, valluvar colony	requester	gokul.ap.2402@gmail.com	1345956789
A Positive	11, Race course road, Madurai	Ramu	ramu@outlook.com	7448610486

