

Assignment -2

Connect with Database Assignment

Assignment Date	18 october 2022
Student Name	Preethi R
Student Roll Number	510519205015
Maximum Marks	2 Marks

Question-1:

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

The screenshot shows a database IDE interface. The top pane displays a SQL script to create a table named 'USER' with columns: EMAIL VARCHAR(30), USERNAME CHAR(20), ROLLNO INT, and PASSWORD VARCHAR(100). The bottom pane shows the 'History' tab with a table listing the execution of the script. Below this, the 'Table View' for 'PHB43134.USER' is shown, displaying the table's structure with columns: EMAIL, USERNAME, ROLLNO, and PASSWORD.

Script	Date	Status	Runtime
Untitled - 1	Oct 9, 2022 1:14:27 PM	✓ 1	0.078 s
CREATE TABLE USER (EMAIL VARCHAR(30), USERNAME CHAR(20), ROLLNO INT, PASSWOR...		✓	0.078 s

EMAIL	USERNAME	ROLLNO	PASSWORD
-------	----------	--------	----------

2. Perform UPDATE, DELETE Queries with User table

The screenshot shows the same database IDE interface. The top pane displays two SQL scripts: an INSERT statement for a user with email 'abc@gmail.com', username 'RAM', roll number '6211', and password 'Abc@1bm'; and another INSERT statement for a user with email 'def@gmail.com', username 'sita', roll number '6212', and password 'Def@1bm'. The bottom pane shows the 'History' tab with a table listing the execution of these scripts.

Script	Date	Status	Runtime
Untitled - 1	Oct 14, 2022 10:14:42 AM	✓ 2	0.018 s
INSERT INTO USER VALUES('abc@gmail.com','RAM','6211','Abc@1bm');		✓	0.010 s
INSERT INTO USER VALUES('def@gmail.com','sita','6212','Def@1bm');		✓	0.008 s

Table View:

IBM Db2 on Cloud

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

BYG92981.USER

Back

Export to CSV

USEREMAIL	USERNAME	ROLLNUMBER	PASSWORD
sho@gmail.com	ISACK	6211	Abs@qsm
sho@gmail.com	sho	6212	Sho@qsm

UPDATE:

IBM Db2 on Cloud

Run selected

```

1 ALTER USER VALUES ('sho@gmail.com', 'KAM', '6211', 'Abs@qsm');
2 INSERT INTO USER VALUES ('sho@gmail.com', 'sho', '6212', 'Sho@qsm');
3 UPDATE USER SET USERNAME='ISACK' WHERE USEREMAIL='sho@gmail.com';

```

History Results

Find history

Script	Date	Status	Runtime
ALTER USER	04/14/2022 10:22:40 AM	Success	0.002 s
UPDATE USER SET USERNAME='ISACK' WHERE USEREMAIL='sho@gmail.com'	04/14/2022 10:22:40 AM	Success	0.007 s
INSERT INTO	04/14/2022 10:22:40 AM	Success	0.013 s
ALTER USER	04/14/2022 10:22:40 AM	Success	0.002 s

Table View:

IBM Db2 on Cloud

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

BYG92981.USER

Back

Export to CSV

USEREMAIL	USERNAME	ROLLNUMBER	PASSWORD
sho@gmail.com	ISACK	6211	Abs@qsm
sho@gmail.com	sho	6212	Sho@qsm

DELETE:

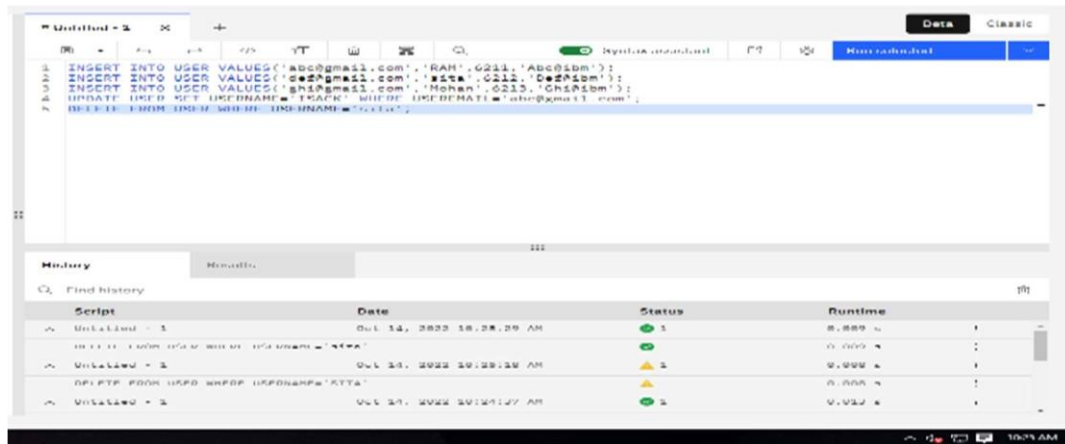
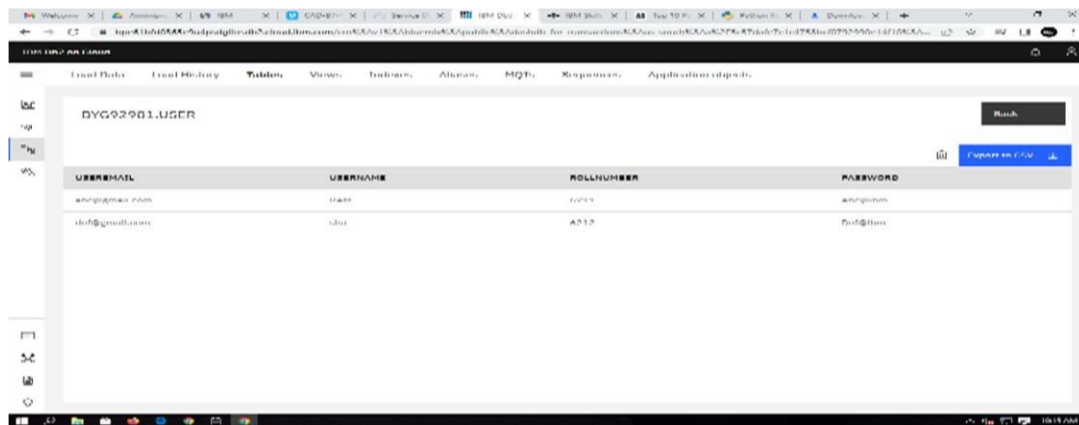


TABLE View:



3. Connect python with db2.

Solution:

```
import ibm_db
```

```
conn = ibm_db.connect("DATABASE=bludb;HOSTNAME= 1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0
nqnrk39u98g.databases.appdomain.cloud;PORT=32286;SECURITY=SSL;SSL
serverCertificate=DigiCertGl
obalRootCA.crt;PROTOCOL=TCPIP;UID= byg92981;PWD=' 9jZpv8EpbeEMaB6i',";")
```

4. Create a flask app with the registration page. Login page and the 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 so the welcome page.

Solution:

```
app.py from flask import Flask, render_template, request, redirect, url_for,
session

import ibm_db
import bcrypt conn
=
ibm_db.connect("DATABASE=bludb;HOSTNAME=;PORT=;SECURITY=SSL;SSLServerCertific
ate=DigiCer tGlobalRootCA.crt;UID=;PWD=\",\"") # url_for('static', filename='style.css')

app = Flask(__name__)
app.secret_key = 'C21FGSBAPOK43K5VSIIDFB2'

@app.route("/",methods=['GET'])
def home():
    if 'email' not in session:
        return redirect(url_for('login'))
    return render_template("home.html",name='Home')

@app.route("/register",methods=['GET','POST'])
def register(): if request.method ==
'POST': email = request.form['email']
username = request.form['username']
rollNo = request.form['rollNo']
password = request.form['password']

if not email or not username or not rollNo or not password:
    return render_template('register.html',error='Please fill all fields')

hash=bcrypt.hashpw(password.encode('utf-8'),bcrypt.gensalt())

query = "SELECT * FROM USER WHERE email=? OR
rollNo=?" stmt = ibm_db.prepare(conn, query)
ibm_db.bind_param(stmt,1,email)
ibm_db.bind_param(stmt,2,rollNo) ibm_db.execute(stmt)
isUser = ibm_db.fetch_assoc(stmt)

if not isUser:
    insert_sql = "INSERT INTO User(username,email,PASSWORD,rollNo) 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, hash) ibm_db.bind_param(prepare_stmt, 4,
rollNo) ibm_db.execute(prepare_stmt)
    return render_template('register.html',success="You can login")
```

```

else:
    return render_template('register.html',error='Invalid
Credentials') return render_template('register.html',name='Home')

@app.route("/login",methods=['GET','POST'])
def login():
    if request.method == 'POST':
        email = request.form['email']
        password = request.form['password']

        if not email or not password:
            return render_template('login.html',error='Please fill all fields')
        query = "SELECT * FROM USER WHERE
email=?" stmt = ibm_db.prepare(conn, query)
        ibm_db.bind_param(stmt,1,email)
        ibm_db.execute(stmt) isUser =
        ibm_db.fetch_assoc(stmt)
        print(isUser,password)
        if not isUser:
            return render_template('login.html',error='Invalid Credentials')

        isPasswordMatch = bcrypt.checkpw(password.encode('utf-
8'),isUser['PASSWORD'].encode('utf-
8'))

        if not isPasswordMatch:
            return render_template('login.html',error='Invalid Credentials')

        session['email'] = isUser['EMAIL']
        return redirect(url_for('home'))

    return render_template('login.html',name='Home')
@app.route('/logout') def
logout():
    session.pop('email', None)
    return redirect(url_for('login'))

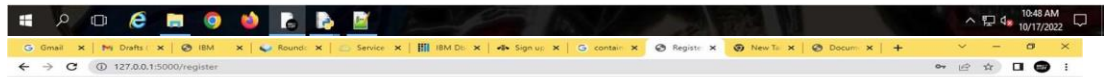
```

OUTPUT:



Register

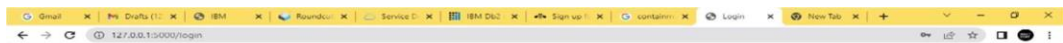
[Don't have an account? Register](#)



Register

You can login

[Don't have an account? Register](#)



Login

[Don't have an account? Register](#)





Database:

