

Assignment -2

Connect with Database Assignment

Assignment Date	09 October 2022
Student Name	Prashna M
Student Roll Number	621319104041
Maximum Marks	2 Marks

Question-1:

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

The screenshot shows the IBM Db2 Cloud console interface. At the top, there's a toolbar with icons for file operations, a 'Run all' button, and a 'Syntax assistant' toggle. The main area displays a SQL script: `CREATE TABLE USER (EMAIL VARCHAR(30), USERNAME CHAR(20), ROLLNO INT, PASSWORD VARCHAR(100));`. Below the script, a 'History' section shows a table with columns: Script, Date, Status, and Runtime. The history entry shows the script was executed on Oct 9, 2022 at 1:14:27 PM, with a status of '1' and a runtime of 0.078 s.

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, PASSWORD...			0.078 s

The screenshot shows the 'Data objects' view in the IBM Db2 Cloud console. It displays the structure of the 'USER' table. The table has four columns: EMAIL, USERNAME, ROLLNO, and PASSWORD. A 'Back' button is visible in the top right corner.

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

2. Perform UPDATE, DELETE Queries with User table

The screenshot shows the IBM Db2 Cloud console interface with a SQL script editor. The script contains an INSERT statement: `INSERT INTO USER VALUES('ram@gmail.com','Ram',12345,'HEll@123');`. The console also shows a 'Data objects' view on the left with a search bar and a list of objects including 'PHB43134'.

EMAIL	USERNAME	ROLLNO	PASSWORD
ram@gmail.com	Ram	12345	HEll@123

Table View:

IBM Db2 on Cloud

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

PHB43134.USER Back

Export to CSV

EMAIL	USERNAME	ROLLNO	PASSWORD
ram@gmail.com	Ram	12345	HEll@123
suresh@gmail.com	Suresh	67890	Sur@123

UPDATE:

*Untitled-1 x + Beta Classic

Syntax assistant Run all

```
1 UPDATE USER SET ROLLNO=87653 WHERE USERNAME='Ram';
```

Table View:

PHB43134.USER Back

Export to CSV

EMAIL	USERNAME	ROLLNO	PASSWORD
ram@gmail.com	Ram	87653	HEll@123
suresh@gmail.com	Suresh	67890	Sur@123

DELETE:

*Untitled-1 x + Beta Classic

Syntax assistant Run all

```
1 INSERT INTO USER VALUES('ram@gmail.com','Ram',12345,'HEll@123');
```

TABLE View:

PHB43134.USER				Back
				 Export to CSV 
EMAIL	USERNAME	ROLLNO	PASSWORD	
ram@gmail.com	Ram	87653	HEll@123	

3. Connect python with db2.

Solution:

```
import ibm_db
```

```
conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32733;SECURITY=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;PROTOCOL=TCPIP;UID=phb43134;PWD=mTTnIE45Raj9A0rr", "", "")
```

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;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=;PWD=", "", "")
```

```
# url_for('static', filename='style.css')
```

```
app = Flask(__name_)
```

```
app.secret_key = 'C21FGSBAPOK43K5VSIDFB2'
```

```
@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')

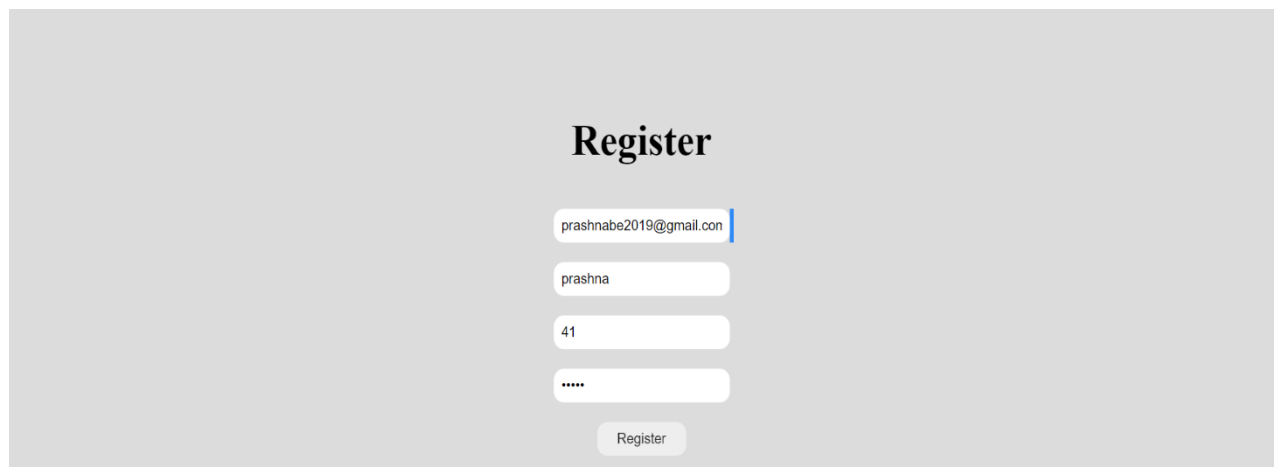
isPasswordMatch = bcrypt.checkpw(password.encode('utf-8'),isUser['PASSWORD'].encode('utf-8'))

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

prashnabe2019@gmail.com

prashna

41

.....

Register

Login

Prashnabe2019@gmail.com

.....

Login

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

