

## Assignment -2

### Connect with Database Assignment

|                     |                 |
|---------------------|-----------------|
| Assignment Date     | 09 October 2022 |
| Student Name        | Rohith V        |
| Student Roll Number | 621319104045    |
| 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 IDE interface. The top panel displays a SQL script for creating a table named 'USER' with columns: EMAIL VARCHAR(30), USERNAME CHAR(20), ROLLNO INT, and PASSWORD VARCHAR(100). The script is executed, and the bottom panel shows the 'History' tab with a table listing the execution details.

| 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 |

Below the history table, the 'Data objects' tab shows the table 'PHB43134.USER' with columns: EMAIL, USERNAME, ROLLNO, and PASSWORD.

2. Perform UPDATE, DELETE Queries with User table

The screenshot shows the IBM Db2 Cloud IDE interface. The top panel displays a SQL script for inserting a new user into the 'USER' table. The script is executed, and the bottom panel shows the 'History' tab with a table listing the execution details.

| Script   | Date                    | Status | Runtime |
|--|-------------------------|--------|---------|
| Untitled - 1   | Oct 27, 2022 2:15:31 PM | ✓ 1    | 0.008 s |
| INSERT INTO USER VALUES('user1@gmail.com', 'user1', '98487', 'user1@123'); |                         | ✓      | 0.008 s |

On the left side, the 'Data objects' tab shows a list of objects, including a table named 'KRR39646'.

Table View:

IBM Db2 on Cloud

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

KRR39646.USER

Export to CSV

| USEREMAIL       | USERNAME | ROLLNUMBER | PASSWORD  |
|-----------------|----------|------------|-----------|
| user1@gmail.com | user1    | 98487      | user1@123 |

UPDATE:

IBM Db2 on Cloud

Data objects My script

Filter objects

KRR39646

Untitled - 1

```
1 INSERT INTO USER VALUES('user1@gmail.com','user1',98487,'user1@123');
2 INSERT INTO USER VALUES('user2@gmail.com','user2',98488,'user2@123');
3 INSERT INTO USER VALUES('user3@gmail.com','user3',98489,'user3@123');
4 UPDATE USER SET USERNAME = 'user4' WHERE ROLLNUMBER = 98488;
```

History

| Script   | Date                    | Status | Runtime |
|--|-------------------------|--------|---------|
| Untitled - 1   | Oct 27, 2022 2:23:09 PM | 1      | 0.006 s |
| UPDATE USER SET USERNAME = 'user4' WHERE ROLLNUMBER = 98488      |                         |        | 0.006 s |
| Untitled - 1   | Oct 27, 2022 2:16:30 PM | 2      | 0.011 s |
| INSERT INTO USER VALUES('user2@gmail.com','user2','98488','us... |                         |        | 0.006 s |
| INSERT INTO USER VALUES('user3@gmail.com','user3','98489','us... |                         |        | 0.005 s |

Table View:

IBM Db2 on Cloud

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

KRR39646.USER

Export to CSV

| USEREMAIL       | USERNAME | ROLLNUMBER | PASSWORD  |
|-----------------|----------|------------|-----------|
| user1@gmail.com | user1    | 98487      | user1@123 |
| user2@gmail.com | user4    | 98488      | user2@123 |
| user3@gmail.com | user3    | 98489      | user3@123 |

DELETE:

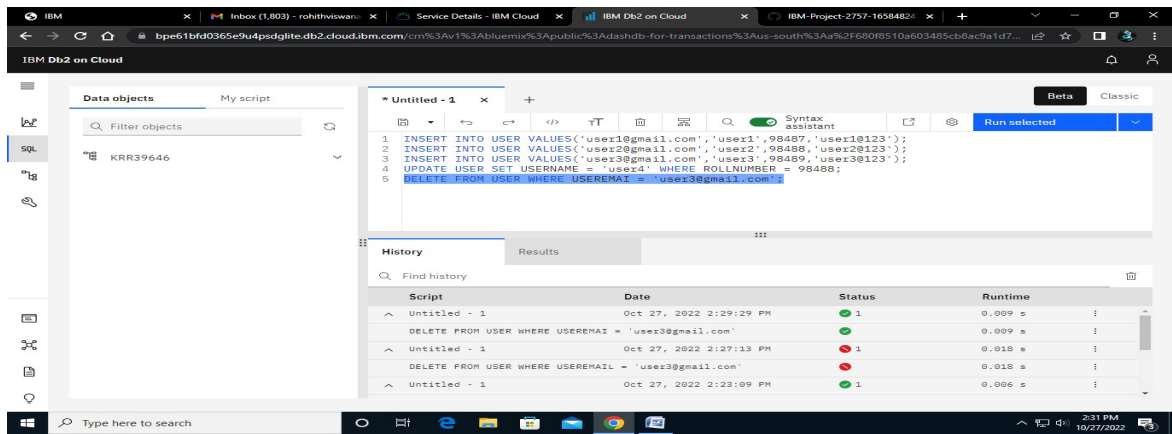
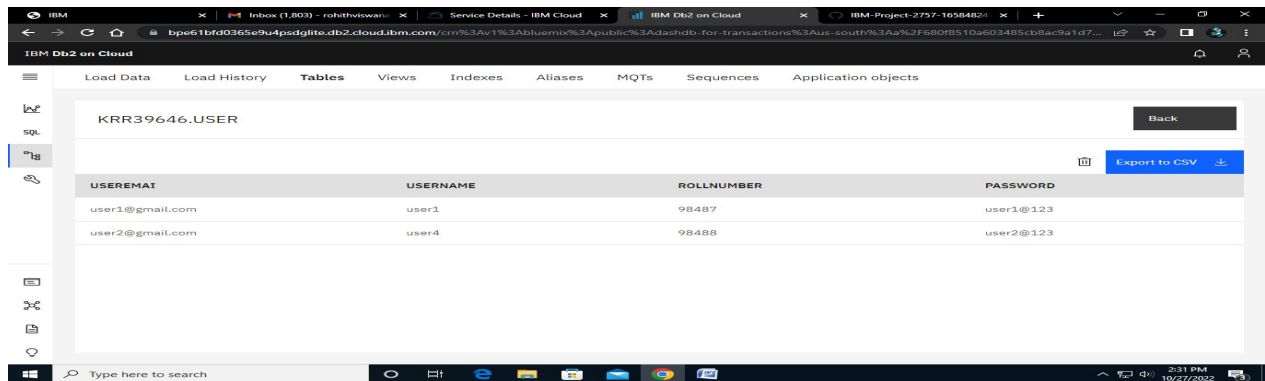


TABLE View:



### 3. Connect python with db2.

Solution:

```
import ibm_db
conn = ibm_db.connect("DATABASE=bludb;HOSTNAME= 1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n4lcmd0
nqnrk39u98g.databases.appdomain.cloud;PORT=30875;SECURITY=SSL;SSLS    erverCertificate=DigiCertGl
obalRootCA.crt;PROTOCOL=TCPIP;UID= krr39646 ;PWD='GbTu7pgBNN76LmGi',' ',' ')
```

### 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=DigiCer
tGlobalRootCA.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)
</pre

```

```

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

Register

{{success}}

{{error}}

[Already have an account? Login](#)



## Login

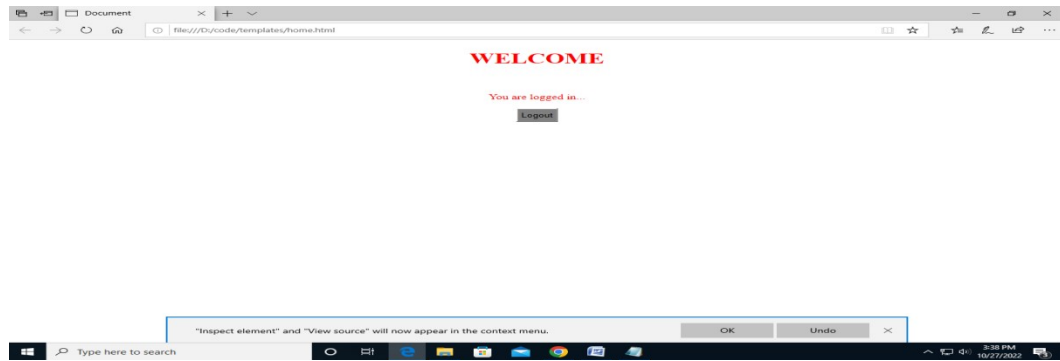
Login

{{success}}

{{error}}

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





## Database:

