

ASSIGNMENT 2

Database connection

Date	12 October 2022
Student Name	Kowsik v
Student Roll no	621319104028
Maximum Marks	2 Marks

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

The screenshot shows the IBM Db2 on Cloud interface. The main editor displays the following SQL script:

```
1 CREATE TABLE user_details(  
2     sl_no INT GENERATED BY DEFAULT AS IDENTITY NOT NULL,  
3     user_name VARCHAR(150) NOT NULL,  
4     user_email VARCHAR(255) NOT NULL,  
5     roll_no INT NOT NULL,  
6     password VARCHAR(100),  
7     PRIMARY KEY (sl_no)  
8 );  
9
```

The History panel at the bottom shows the execution of the script:

Script	Date	Status	Runtime
ASSIGNMENT_2	Oct 16, 2022 3:52:59 PM	✓ 1	0.161 s
CREATE TABLE user_details(sl_no INT GENERATED BY DEFAULT AS IDENTITY NOT NULL, us...		✓	0.161 s

Inserting values :

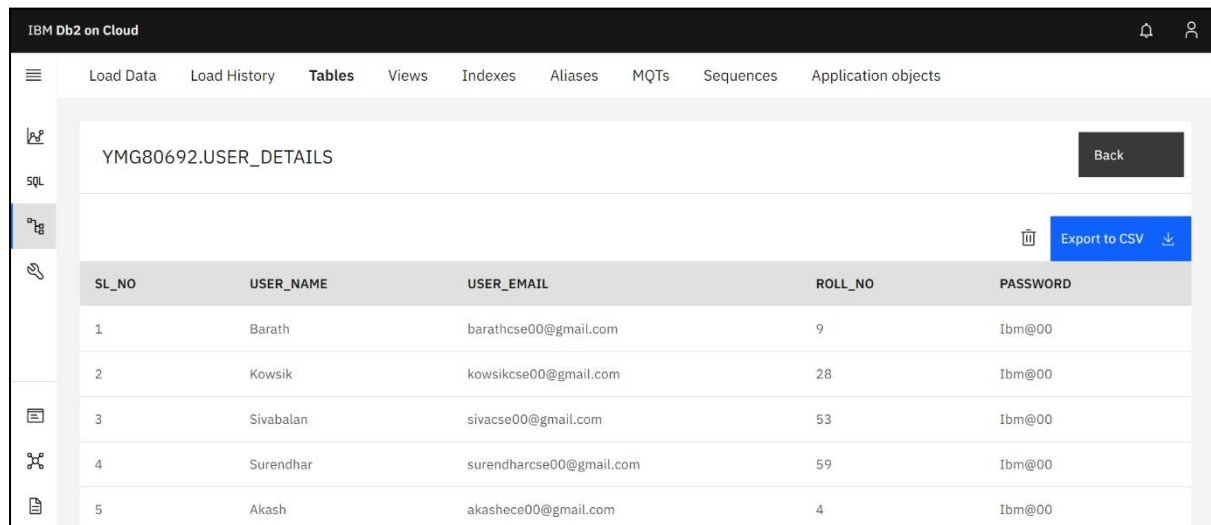
The screenshot shows the IBM Db2 on Cloud interface with the following SQL script in the editor:

```
9  
10 insert into user_details(user_name,user_email,roll_no,password) values('Barath','barathcse00@gmail.com',09,'Ibm@00');  
11 insert into user_details(user_name,user_email,roll_no,password) values('kowsik','kowsikcse00@gmail.com',28,'Ibm@00');  
12 insert into user_details(user_name,user_email,roll_no,password) values('Sivabalan','sivacse00@gmail.com',53,'Ibm@00');  
13 insert into user_details(user_name,user_email,roll_no,password) values('Surendhar','surendharcse00@gmail.com',59,'Ibm@00');  
14 insert into user_details(user_name,user_email,roll_no,password) values('Akash','akashece00@gmail.com',04,'Ibm@00');
```

The History panel shows the execution of these insert statements:

Script	Date	Status	Runtime
ASSIGNMENT_2	Oct 16, 2022 3:54:21 PM	✓ 5	0.052 s
insert into user_details(user_name,user_email,roll_no,password) values('Barath','...		✓	0.013 s
insert into user_details(user_name,user_email,roll_no,password) values('kowsik','...		✓	0.009 s
insert into user_details(user_name,user_email,roll_no,password) values('Sivabalan...		✓	0.011 s
insert into user_details(user_name,user_email,roll_no,password) values('Surendhar...		✓	0.008 s

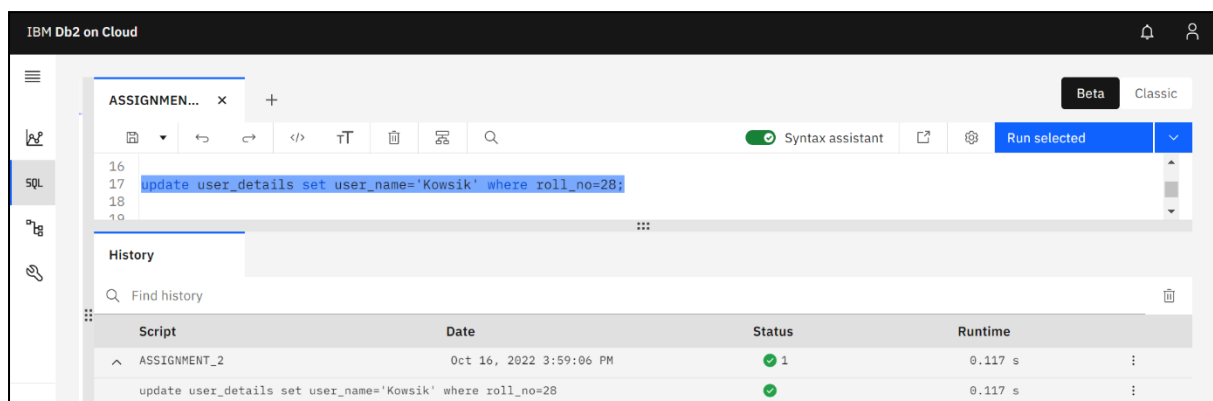
User_details table:



The screenshot shows the IBM Db2 on Cloud interface. The top navigation bar includes 'Load Data', 'Load History', 'Tables', 'Views', 'Indexes', 'Aliases', 'MQTs', 'Sequences', and 'Application objects'. The 'Tables' tab is selected, and the table 'YMG80692.USER_DETAILS' is displayed. A 'Back' button is in the top right. Below the table name, there is an 'Export to CSV' button. The table structure is as follows:

SL_NO	USER_NAME	USER_EMAIL	ROLL_NO	PASSWORD
1	Barath	barathcse00@gmail.com	9	Ibm@00
2	Kowsik	kowsikcse00@gmail.com	28	Ibm@00
3	Sivabalan	sivacse00@gmail.com	53	Ibm@00
4	Surendhar	surendharcse00@gmail.com	59	Ibm@00
5	Akash	akasecse00@gmail.com	4	Ibm@00

2. Perform update & delete queries with the table

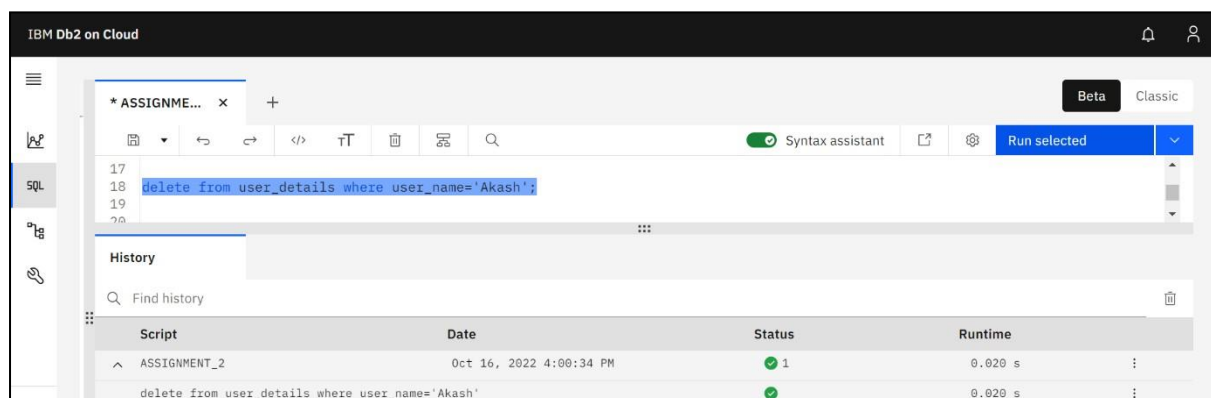


The screenshot shows the IBM Db2 on Cloud interface with the 'SQL' editor. The query being executed is:

```
update user_details set user_name='Kowsik' where roll_no=28;
```

The 'History' tab is open, showing the execution details of the query:

Script	Date	Status	Runtime
ASSIGNMENT_2	Oct 16, 2022 3:59:06 PM	✓ 1	0.117 s
update user_details set user_name='Kowsik' where roll_no=28		✓	0.117 s



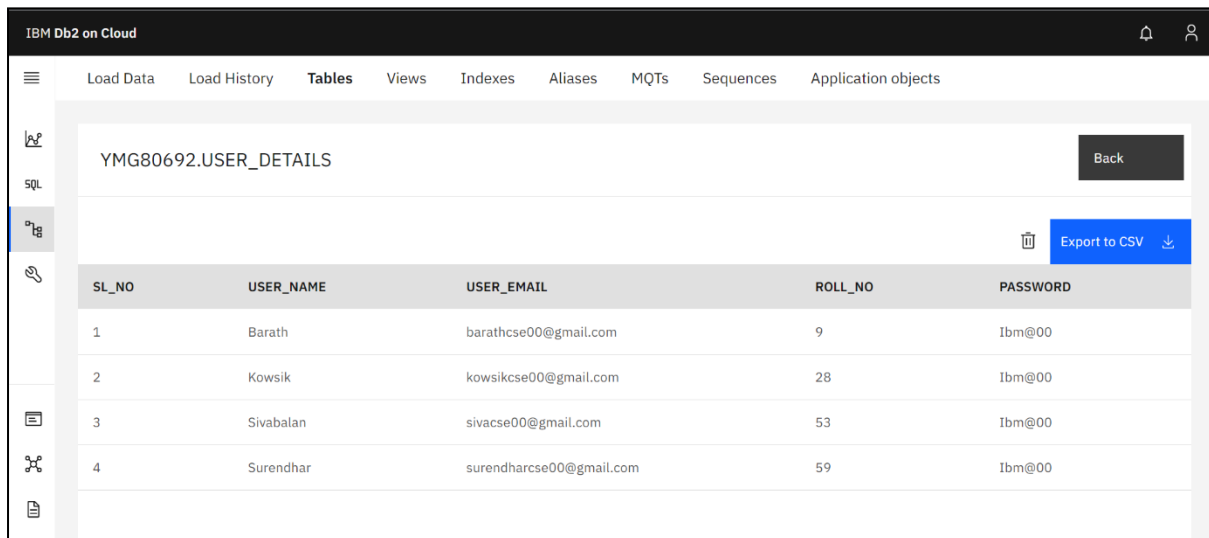
The screenshot shows the IBM Db2 on Cloud interface with the 'SQL' editor. The query being executed is:

```
delete from user_details where user_name='Akash';
```

The 'History' tab is open, showing the execution details of the query:

Script	Date	Status	Runtime
ASSIGNMENT_2	Oct 16, 2022 4:00:34 PM	✓ 1	0.020 s
delete from user_details where user_name='Akash'		✓	0.020 s

Updated table:



The screenshot shows the IBM Db2 on Cloud web interface. The top navigation bar includes 'Load Data', 'Load History', 'Tables', 'Views', 'Indexes', 'Aliases', 'MQTs', 'Sequences', and 'Application objects'. The 'Tables' tab is selected, and the table 'YMG80692.USER_DETAILS' is displayed. The table has five columns: 'SL_NO', 'USER_NAME', 'USER_EMAIL', 'ROLL_NO', and 'PASSWORD'. There are four rows of data. To the right of the table, there are buttons for 'Back', 'Export to CSV', and a download icon.

SL_NO	USER_NAME	USER_EMAIL	ROLL_NO	PASSWORD
1	Barath	barathcse00@gmail.com	9	Ibm@00
2	Kowsik	kowsikcse00@gmail.com	28	Ibm@00
3	Sivabalan	sivacse00@gmail.com	53	Ibm@00
4	Surendhar	surendharcse00@gmail.com	59	Ibm@00

3. Connect python code to db2

```
import ibm_db
```

```
conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=98538591-7217-4024-b0278baa776ffad1.c3n41cmd0nqnrk39u98.databases.appdomain.cloud;PORT=30875;SECURITY=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;PROTOCOL=TCPIP;UID=ymg80692;PWD=dS5CZPeX9CN20vpY", "", "")
```

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

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

```
import ibm_db
```

```
import bcrypt
```

```
conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=98538591-7217-4024-b0278baa776ffad1.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud;PORT=30875;SECURITY=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;PROTOCOL=TCPIP;UID=ymg80692;PWD=dS5CZPeX9CN20vpY", "", "")
```

```

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

app = Flask(_name_)
app.secret_key = b'_5#y2L"F4Q8z\n\xec]/'

@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(EMAIL, USERNAME, ROLLNO, PASSWORD) VALUES
            (?, ?, ?, ?)"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prepare_stmt, 1, email)
            ibm_db.bind_param(prepare_stmt, 2, username)
            ibm_db.bind_param(prepare_stmt, 3, rollNo)
            ibm_db.bind_param(prepare_stmt, 4, hash)
            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'))

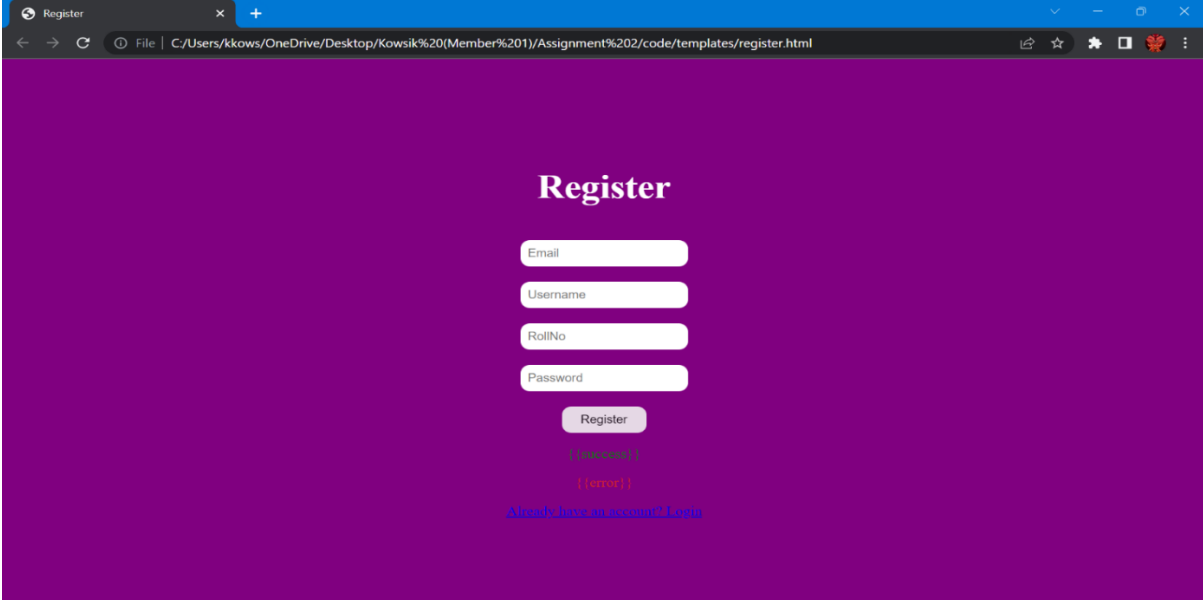
    return render_template('login.html',name='Home')

@app.route('/logout')
def logout():
    session.pop('email', None)
    return redirect(url_for('login'))
if __name__ == "__main__":
    app.run(debug=True)

```

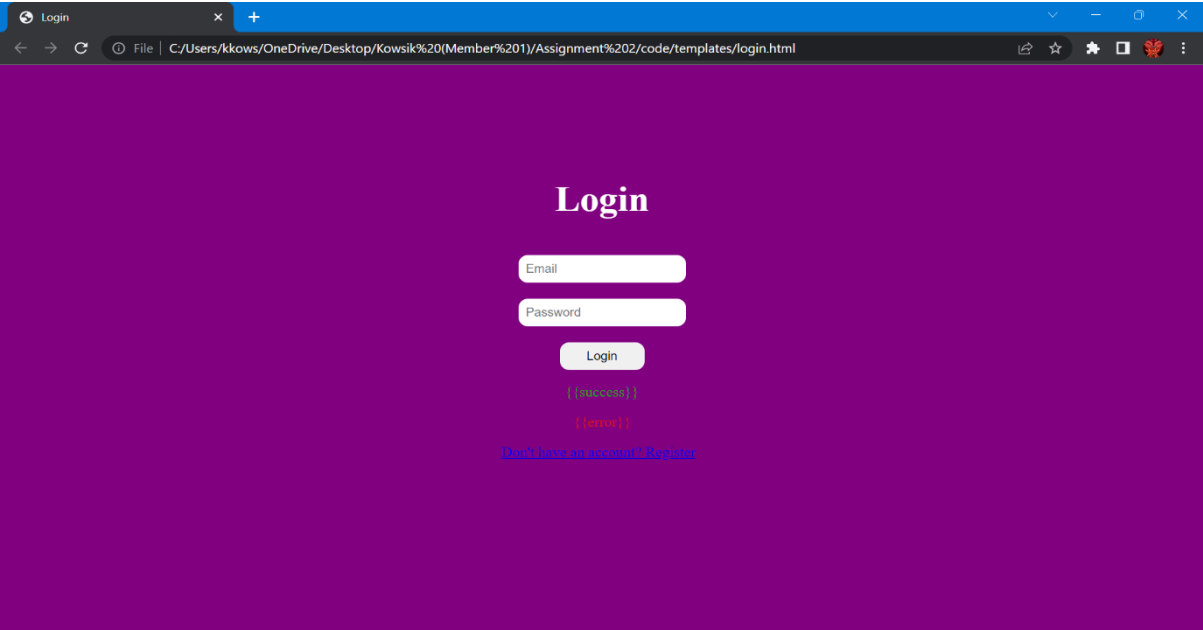
Output:

Registering:



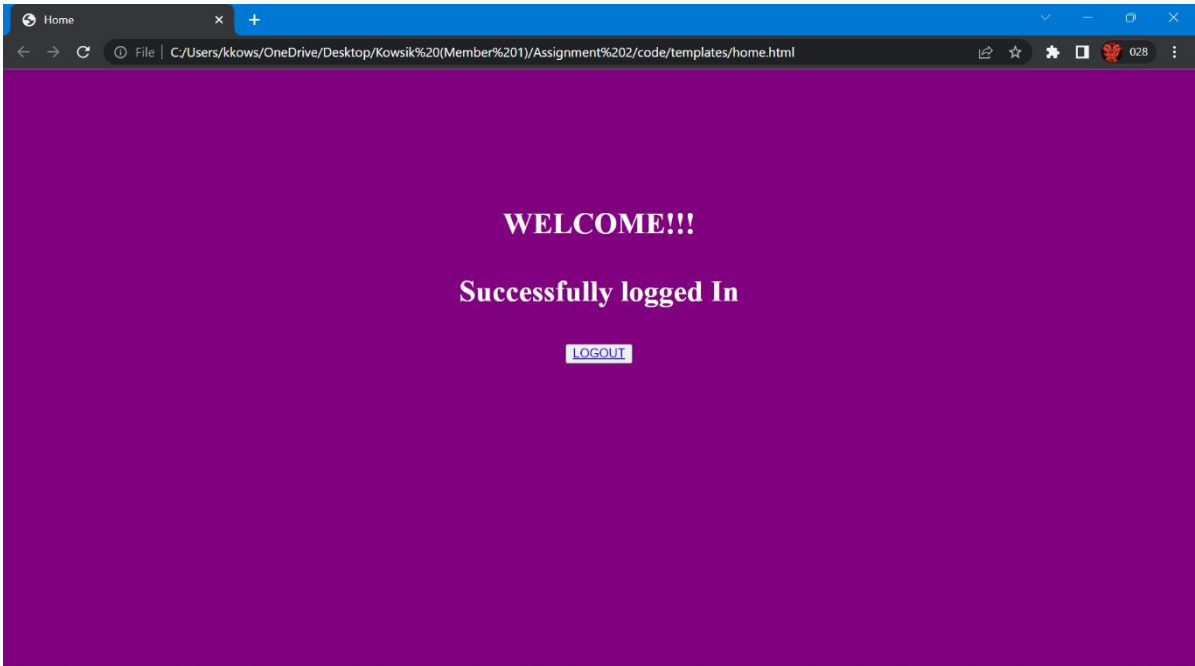
The screenshot shows a web browser window with the title 'Register'. The address bar displays the file path: `C:/Users/kkows/OneDrive/Desktop/Kowsik%20(Member%201)/Assignment%202/code/templates/register.html`. The page has a solid purple background. In the center, there is a registration form with the title 'Register' in a large, white, serif font. Below the title are four white input fields with black text labels: 'Email', 'Username', 'RollNo', and 'Password'. A white 'Register' button is positioned below the 'Password' field. Below the button, there are two lines of placeholder text: `{{success}}` in green and `{{error}}` in red. At the bottom of the form, there is a blue link that reads 'Already have an account? Login'.

Logging in:



The screenshot shows a web browser window with the title 'Login'. The address bar displays the file path: `C:/Users/kkows/OneDrive/Desktop/Kowsik%20(Member%201)/Assignment%202/code/templates/login.html`. The page has a solid purple background. In the center, there is a login form with the title 'Login' in a large, white, serif font. Below the title are two white input fields with black text labels: 'Email' and 'Password'. A white 'Login' button is positioned below the 'Password' field. Below the button, there are two lines of placeholder text: `{{success}}` in green and `{{error}}` in red. At the bottom of the form, there is a blue link that reads 'Don't have an account? Register'.

Home page:



Database:

