

Assignment	2
Roll No	1901016
Name	Arjun RU

1. Create a User table with email, username, roll number, password.
2. Perform UPDATE,DELETE Queries with user table.
3. Connect python code to db2.
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 the database and navigate to the login page authenticate user username and password. If the user is valid, show the welcome page.

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

CREATE Statement:

```
CREATE TABLE "LLG47061".User ( Name VARCHAR(32) NOT NULL ,RollNo VARCHAR(32) NOT NULL ,Email VARCHAR(32) NOT NULL )
```

OUTPUT:

The screenshot shows a SQL IDE interface. At the top, there's a toolbar with icons for file operations, code editing, and execution. A 'Syntax assistant' toggle is visible. Below the toolbar, a SQL statement is entered in the editor: `CREATE TABLE "LLG47061".User (Name VARCHAR(32) NOT NULL ,RollNo VARCHAR(32) NOT NULL ,Email VARCHAR(32) NOT NULL)`. Below the editor, there are tabs for 'History' and 'Results'. The 'History' tab is active, showing a table with columns: Script, Date, Status, and Runtime. The table contains one entry for the executed statement, which is marked as successful (green checkmark) and took 0.052 s to run.

Script	Date	Status	Runtime
Untitled - 1	Nov 11, 2022 12:30:52 PM	✓ 1	0.052 s
CREATE TABLE "LLG47061".User (Name VARCHAR(32) NOT NULL ,RollNo...		✓	0.052 s

2. Perform UPDATE,DELETE Queries with user table.

INSERT Statement:

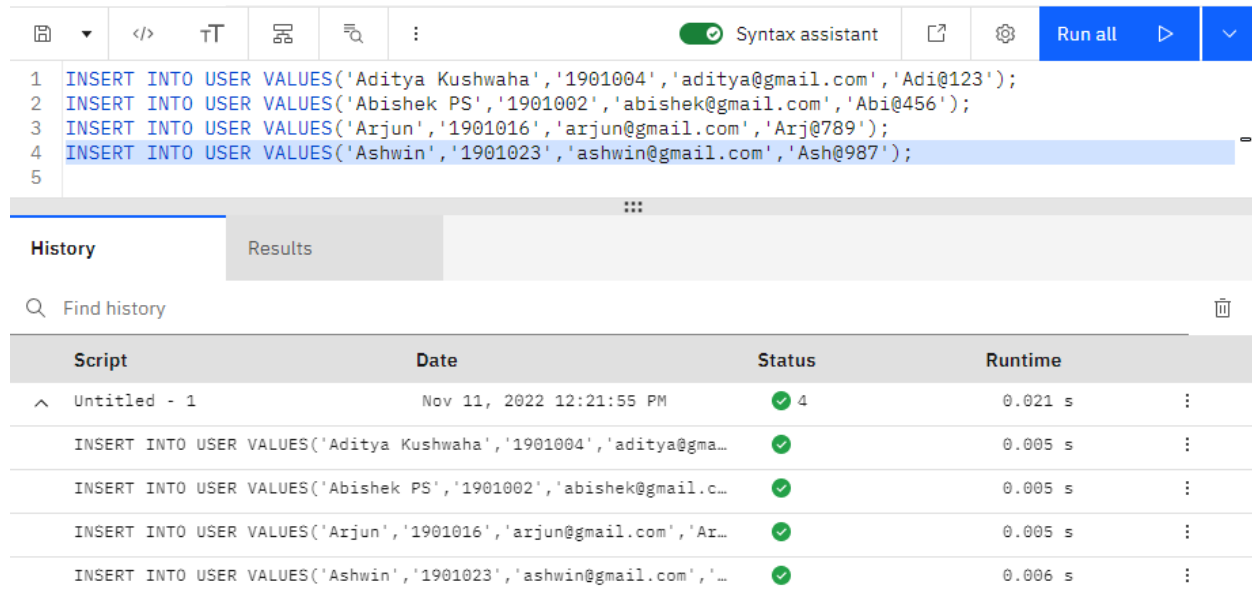
```
INSERT INTO USER VALUES("Aditya Kushwaha","1901004","aditya@gmail.com","Adi@123");
INSERT INTO USER VALUES("Abishek
```

PS","1901002","abishek@gmail.com","Abi@456");

INSERT INTO USER VALUES("Arjun","1901016","arjun@gmail.com","Arj@789");

INSERT INTO USER VALUES("Ashwin","1901023","ashwin@gmail.com","Ash@987");

OUTPUT:



The screenshot shows a SQL IDE interface with a code editor and a history table. The code editor contains four INSERT statements. The history table shows the execution of these statements, all of which were successful.

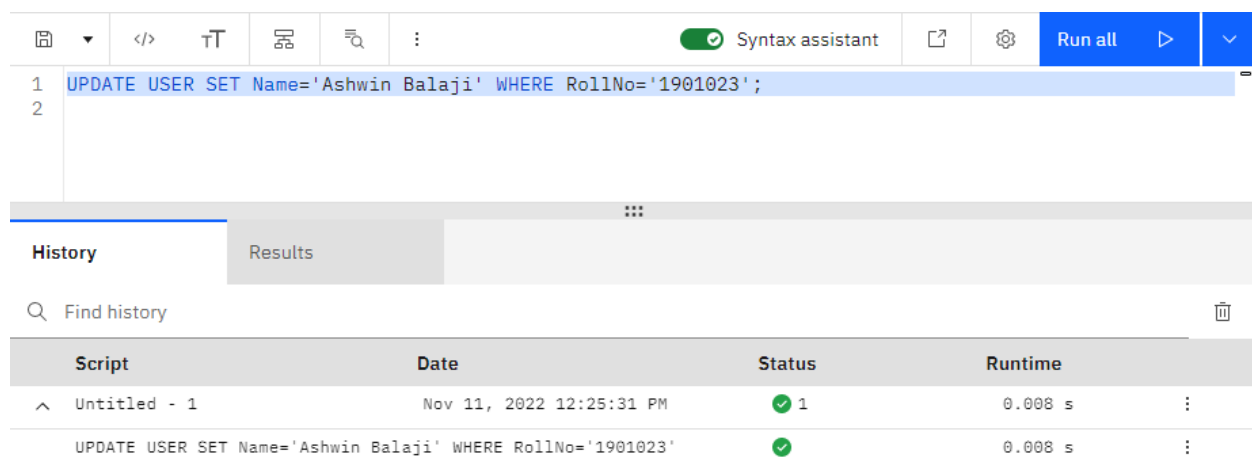
```
1 INSERT INTO USER VALUES('Aditya Kushwaha','1901004','aditya@gmail.com','Adi@123');
2 INSERT INTO USER VALUES('Abishek PS','1901002','abishek@gmail.com','Abi@456');
3 INSERT INTO USER VALUES('Arjun','1901016','arjun@gmail.com','Arj@789');
4 INSERT INTO USER VALUES('Ashwin','1901023','ashwin@gmail.com','Ash@987');
```

Script	Date	Status	Runtime
Untitled - 1	Nov 11, 2022 12:21:55 PM	✓ 4	0.021 s
INSERT INTO USER VALUES('Aditya Kushwaha','1901004','aditya@gmail.com','Adi@123');		✓	0.005 s
INSERT INTO USER VALUES('Abishek PS','1901002','abishek@gmail.com','Abi@456');		✓	0.005 s
INSERT INTO USER VALUES('Arjun','1901016','arjun@gmail.com','Arj@789');		✓	0.005 s
INSERT INTO USER VALUES('Ashwin','1901023','ashwin@gmail.com','Ash@987');		✓	0.006 s

UPDATE Statement:

UPDATE USER SET Name='Ashwin Balaji' WHERE RollNo='1901023';

OUTPUT:



The screenshot shows a SQL IDE interface with a code editor and a history table. The code editor contains an UPDATE statement. The history table shows the execution of this statement, which was successful.

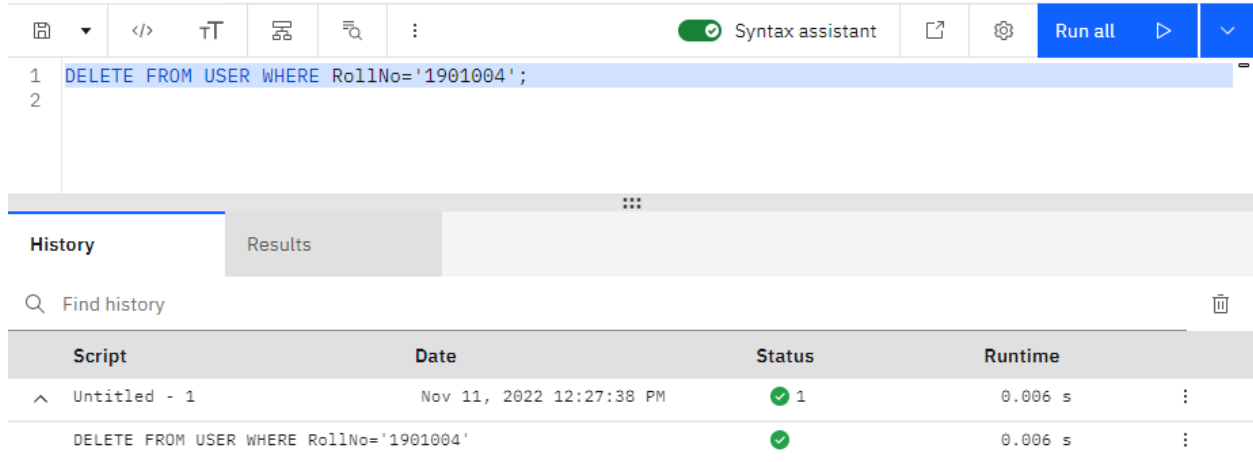
```
1 UPDATE USER SET Name='Ashwin Balaji' WHERE RollNo='1901023';
2
```

Script	Date	Status	Runtime
Untitled - 1	Nov 11, 2022 12:25:31 PM	✓ 1	0.008 s
UPDATE USER SET Name='Ashwin Balaji' WHERE RollNo='1901023';		✓	0.008 s

DELETE Statement:

DELETE FROM USER WHERE RollNo='1901004';

OUTPUT:



The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, a code editor, a syntax assistant (checked), a settings icon, and buttons for 'Run all', 'Play', and a dropdown. The code editor contains the following SQL statement:

```
1 DELETE FROM USER WHERE RollNo='1901004';
2
```

Below the code editor, there are tabs for 'History' and 'Results'. The 'History' tab is active, showing a search bar and a table of executed scripts.

Script	Date	Status	Runtime
^ Untitled - 1	Nov 11, 2022 12:27:38 PM	✓ 1	0.006 s
DELETE FROM USER WHERE RollNo='1901004'		✓	0.006 s

3. Connect python code to db2.

PROGRAM:

```
import ibm_db
dsn_hostname =
"9938aec0-8105-433e-8bf9-0fbb7e483086.c1ogj3sd0tgtu0lqde00.databases.appdomai
n.cloud"
dsn_uid = "lg47061"
dsn_pwd = "SGk1gUnQkd6ECxfp"
dsn_driver = "{IBM DB2 ODBC DRIVER}"
dsn_database = "bludb"
dsn_port = "32459"
dsn_protocol = "TCPIP"
dsn_security = "SSL"
dsn = (
"DRIVER={0};"
"DATABASE={1};"
"HOSTNAME={2};"
"PORT={3};"
"PROTOCOL={4};"
```

```

"UID={5};"
"PWD={6};"
"SECURITY={7};"
).format(
    dsn_driver,
    dsn_database,
    dsn_hostname,
    dsn_port,
    dsn_protocol,
    dsn_uid,
    dsn_pwd,
    dsn_security,
)
print(dsn)
try:
    conn = ibm_db.connect(dsn, "", "")
    print ("Connected to database: ", dsn_database, "as user: ", dsn_uid, "on host: ",
    dsn_hostname)
except:
    print ("Unable to connect: ", ibm_db.conn_errormsg() )

```

OUTPUT:

```

DRIVER={IBM DB2 ODBC DRIVER};DATABASE=bludb;HOSTNAME=9938aec0-8105-433e-8bf9-0fbb7e483086.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;
PORT=32459;PROTOCOL=TCPIP;UID=1lg47061;PWD=SGk1gUnQkd6ECxfg;SECURITY=SSL;
Connected to database:  bludb as user:  1lg47061 on host:  9938aec0-8105-433e-8bf9-0fbb7e483086.c1ogj3sd0tgtu0lqde00.databases.appdomain.
cloud

```

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 the database and navigate to the login page authenticate user username and password. If the user is valid, show the welcome page.

PROGRAM:

app.py:

```

from flask import Flask, render_template,request,session,redirect
import ibm_db
app = Flask(__name__)

```

```

app.secret_key='a'
dsn_hostname =
"9938aec0-8105-433e-8bf9-0fbb7e483086.c1ogj3sd0tgtu0lqde00.databases.appdomai
n.cloud"
dsn_uid = "llg47061"
dsn_pwd = "SGk1gUnQkd6ECxfg"
dsn_driver = "{IBM DB2 ODBC DRIVER}"
dsn_database = "bludb"
dsn_port = "32459"
dsn_protocol = "TCPIP"
dsn_security = "SSL"
dsn = (
"DRIVER={0};"
"DATABASE={1};"
"HOSTNAME={2};"
"PORT={3};"
"PROTOCOL={4};"
"UID={5};"
"PWD={6};"
"SECURITY={7};"
).format(
dsn_driver,
dsn_database,
dsn_hostname,
dsn_port,
dsn_protocol,
dsn_uid,
dsn_pwd,
dsn_security,
)
try:
conn = ibm_db.connect(dsn, "", "")
print ("Connected to database: ", dsn_database, "as user: ", dsn_uid, "on host: ",
dsn_hostname)
@app.route("/",methods=['GET','POST'])
@app.route("/register",methods=['GET','POST'])
def register():
msg = None
if request.method == 'POST':
name=request.form['name']

```

```

rollno=request.form['rollno']
email=request.form['email']
password=request.form['password']
select_sql = "Select * from User where email=?"
stmt = ibm_db.prepare(conn,select_sql)
ibm_db.bind_param(stmt,1,email)
ibm_db.execute(stmt)
user = ibm_db.fetch_assoc(stmt)
if user:
    msg = "User with this email already exists!"
    return render_template("register.html",msg = msg)
else:
    insert_sql = "Insert into User Values (?, ?, ?, ?)"
    stmt = ibm_db.prepare(conn,insert_sql)
    ibm_db.bind_param(stmt,1,name)
    ibm_db.bind_param(stmt,2,rollno)
    ibm_db.bind_param(stmt,3,email)
    ibm_db.bind_param(stmt,4,password)
    ibm_db.execute(stmt)
    return redirect("/login")
    return render_template("register.html")
@app.route("/login",methods=['GET','POST'])
def login():
    msg = None
    if request.method == 'POST':
        email=request.form['email']
        password=request.form['password']
        select_sql = "Select * from User where email=?"
        stmt = ibm_db.prepare(conn,select_sql)
        ibm_db.bind_param(stmt,1,email)
        ibm_db.execute(stmt)
        user = ibm_db.fetch_assoc(stmt)
        if not user:
            msg = "User with this email does not exist!"
            return render_template("login.html",msg = msg)
        elif user['PASSWORD'] != password:
            msg = "Incorrect Passowrd!"
            return render_template("login.html",msg = msg)
        else:
            session['loggedin'] = True

```

```

session['id'] = user['EMAIL']
session['username'] = user['NAME']
return redirect('/welcome')
return render_template("login.html")
@app.route("/logout")
def logout():
    session['loggedin'] = False
    session['id'] = ""
    session['username'] = ""
    return render_template("login.html",msg="Logged out successfully!")
@app.route("/welcome")
def welcome():
    return render_template("welcome.html",)
except:
    print ("Unable to connect: ", ibm_db.conn_errormsg() )
TEMPLATES:
register.html:
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<link rel="stylesheet" href="{{url_for('static', filename='css/style.css')}}">
<title>Registration Form</title>
</head>
<body>
<div class="container">
<div class="header">
<h1>Sign In Form</h1>
</div>
{% if msg %}
<div class="message">
<span>{{msg}}</span>
</div>
{% endif %}
<div class="form-container">
<form action="/register" method="POST">
<div class="form-field-container">
<label for="email">Name</label>
<input type="text" name="name" required>

```

```

</div>
<div class="form-field-container">
<label for="rollno">Roll No</label>
<input type="text" name="rollno" required>
</div>
<div class="form-field-container">
<label for="email">Email</label>
<input type="email" name="email" required>
</div>
<div class="form-field-container">
<label for="password">Password</label>
<input type="password" name="password" required>
</div>
<div class="form-field-container">
<button type="submit">Submit</button>
<button type="reset">Cancel</button>
</div>
</form>
</div>
<div class="footer">
<span>Already have an account? <a href="/login">Login</a></span>
</div>
</div>
</body>
</html>

```

login.html:

```

<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<link rel="stylesheet" href="{{url_for('static', filename='css/style.css')}}">
<title>Login Form</title>
</head>
<body>
<div class="container">
<div class="header">
<h1>Login Form</h1>
</div>
{% if msg %}

```



```

<div class="message">
<span>{{msg}}</span>
</div>
{% endif %}
<div class="form-container">
<form action="/login" method="POST">
<div class="form-field-container">
<label for="email">Email</label>
<input type="text" name="email" required>
</div>
<div class="form-field-container">
<label for="password">Password:</label>
<input type="password" name="password" required>
</div>
<div class="form-field-container">
<button type="submit">Submit</button>
<button type="reset">Cancel</button>
</div>
</form>
</div>
<div class="footer">
<span>Don't have an account? <a href="/register">Sign Up</a></span>
</div>
</div>
</body>
</html>

```

welcome.html:

```

<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Welcome Page</title>
</head>
<body>
<h1>Welcome {{session['username']}}!</h1>
</body>
</html>

```

STATIC:

style.css

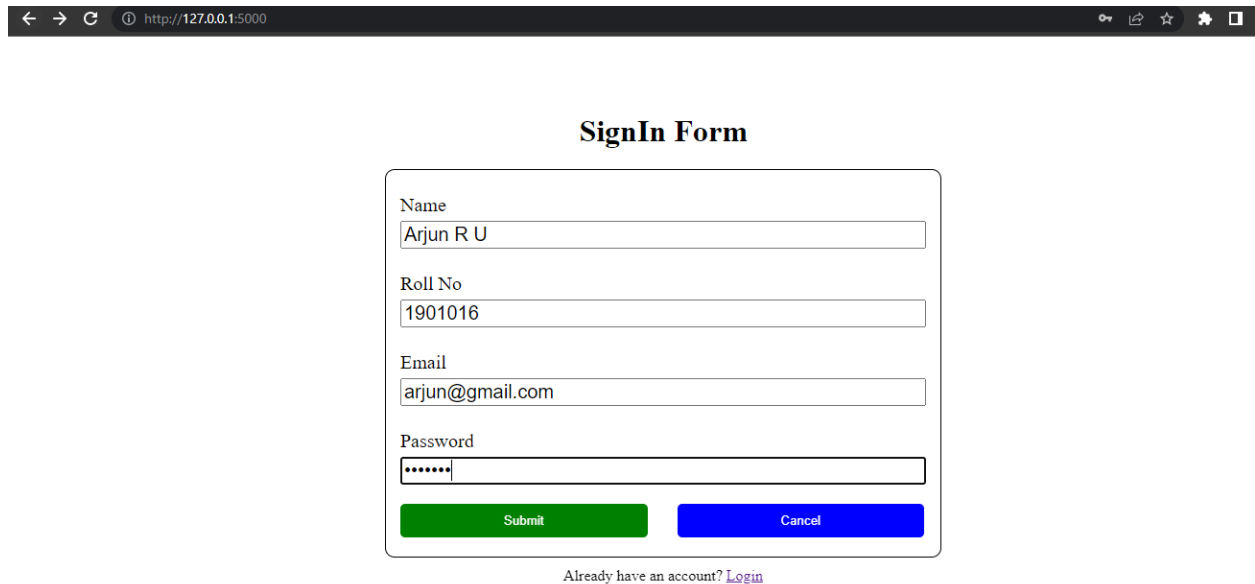
```
.header{
text-align: center;
}
.container{
margin: 80px auto;
width: 600px;
padding-left: 20px;
padding-right: 20px;
}
.form-container{
margin: 10px;
padding: 10px;
border: 1px solid black;
border-radius: 10px;
}
.form-field-container{
padding: 10px 5px;
}
.form-input-field{
padding: 10px 5px;
}
label{
width: 80px;
display: block;
padding: 5px 0px;
font-size: 20px;
}
input{
width: 100%;
font-size: 20px;
box-sizing: border-box;
}
button{
padding: 10px 20px;
color: white;
border: none;
border-radius: 5px;
width: 47%;
}
button[type=submit]{
```

```
background-color: green;
margin-right: 15px;
}
button[type=reset]{
background-color: blue;
margin-left: 12px;
}
.message{
margin: 0px 10px;
padding: 5px 10px;
border-radius: 5px;
text-align: center;
}
.footer{
text-align: center;
}
```

OUTPUT:

i) Registration:

Default loading page ("/" or "/register"):

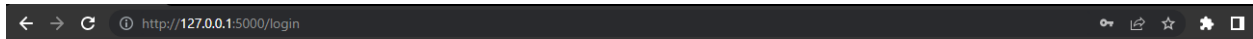


The screenshot shows a web browser window with the address bar displaying "http://127.0.0.1:5000". The main content area features a "SignIn Form" with the following elements:

- Name:** A text input field containing "Arjun R U".
- Roll No:** A text input field containing "1901016".
- Email:** A text input field containing "arjun@gmail.com".
- Password:** A text input field with masked characters "*****".
- Buttons:** A green "Submit" button and a blue "Cancel" button.
- Footer:** A link that says "Already have an account? [Login](#)".

ii) Login:

After registration, redirected to login page ("/login"):



Login Form

Email

Password:

Submit

Cancel

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

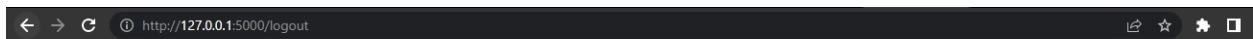
If user is valid:



Welcome Arjun!

[Logout](#)

On logout:



Login Form

Logged out successfully!

Email

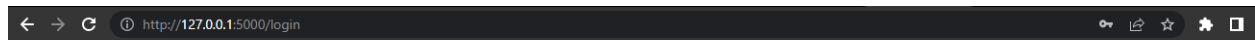
Password:

Submit

Cancel

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

If email not found:



Login Form

User with this email does not exist!

Email

Password:

Submit

Cancel

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

If wrong password:



Login Form

Incorrect Passowrd!

Email

Password:

Submit

Cancel

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