Assignment-2

|  |  |
| --- | --- |
| Team ID | PNT2022TMID30875 |
| Project Name | SMART FASHION  RECOMMENDER APPLICATION |

Question 1: Create User table with username , email , roll number and password. QUERY:

CREATE TABLE USERS( empId INTEGER PRIMARY KEY, ROLL NO TEXT NOT NULL,

USERNAME TEXT NOT NULL,

EMAIL ID TEXT NOT NULL,

PASSWORD TEXT NOT NULL

);

INSERT INTO USERS VALUES(01,'JANANI SK','skjanani107@gmail.com’,'janani@');

INSERT INTO USERS VALUES(03,'INDHUMATHI

S','indhumathisivaraman21@gmail.com','indhumathi@');

INSERT INTO USERS VALUES(02,'ABINAYASRI M','abinayasri2196@gmail.com','abinayasri@');

INSERT INTO USERS VALUES(04,'LOGESHWARI

K','klogeshwari18032002@gmail.com','logeshwari@');

SELECT \* FROM USERS;



2. Perform UPDATE, DELETE Queries with user table.

Updation:

UPDATE USERS

SET Email='skjanu@gmail.com'

WHERE RollNumber=01;

UPDATE USERS

SET Password='m\_abi’

WHERE RollNumber=02, SELECT \* FROM USERS;



Deletion:

DELETE FROM USERS

WHERE RollNumber=03;



3. Connect python code to db2.

import ibm\_db

conn=ibm\_db.connect("DATABASE=bludb;AUTHENTICATION=SERVER;HOSTNAME=19a

f6446

-6171-4641-8aba-

9dcff8e1b6ff.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=30699;SECURITY=SS L;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=lzl34218;PWD=3XYDnCQAbEBZID

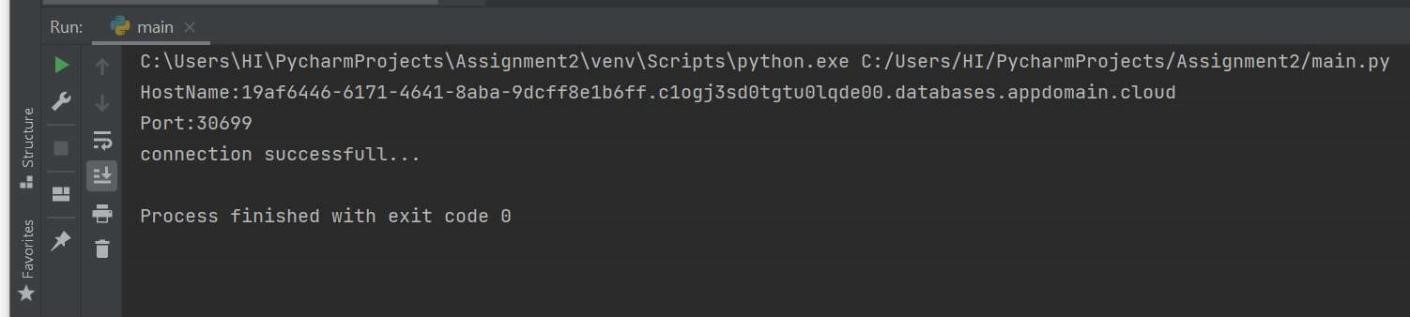
Lb",''

,'')

print(conn)

print("connection successfull...")

Output:



4. Create flask app with registration page, login page and welcome page. app.py from flask import Flask, render\_template, request,redirect import ibm\_db import dbconn app = Flask( name ) @app.route('/') def register():

return render\_template("register.html")

@app.route('/login') def login():

return render\_template("login.html")

#welcome page code

@app.route('/welcome') def welcome():

if dbconn.con: sql="SELECT \*

FROM authentication

smt= ibm\_db.exec\_immediate(dbconn.con,sql) res = ibm\_db.fetch\_both(smt) result=[] while res != False: result.append(res) row=len(result) res = ibm\_db.fetch\_both(smt) return render\_template("welcome.html",r=result,rows=row)

#registration page code

@app.route("/signup", methods=['POST']) def signup(): if request.method == 'POST':

name = request.form.get('name') email = request.form.get('email') pwd = request.form.get('pwd') roll

= request.form.get('roll') sql = "INSERT INTO authentication (username,email,password,rollnumber) VALUES

('{0}','{1}','{2}','{3}')"

res = ibm\_db.exec\_immediate(dbconn.con, sql.format(name, email, pwd, roll)) if sql: return

redirect("/login") else: return redirect("/") else:

print("Could'nt store anything...")

#login page code

@app.route("/signin", methods=['POST']) def signin():

if request.method == 'POST': email = request.form.get('email') pwd

= request.form.get('pwd')

sql = "SELECT \* FROM authentication WHERE EMAIL='{0}' AND PASSWORD='{1}'"

smt = ibm\_db.prepare(dbconn.con, sql.format(email,pwd)) ibm\_db.execute(smt) res=ibm\_db.fetch\_assoc(smt) if res:

return redirect("/welcome")

else: return redirect("/login") else:

print("Could'nt store anything...")

#update page code

@app.route("/update/<string:id>", methods=['POST','GET']) def update(id):

if request.method =='POST':

name = request.form.get('name') email = request.form.get('email') pwd = request.form.get('pwd') roll

= request.form.get('roll') sql="UPDATE authentication SET username='{0}',email='{1}',password='{2}',rollnumber='{3}'

WHERE id='{4}'" res = ibm\_db.exec\_immediate(dbconn.con,

sql.format(name,email,pwd,roll,id))

if sql:

return redirect("/welcome")

else: return redirect("/") else:

print("Could'nt store anything...") sql="SELECT \*

FROM authentication WHERE id='{0}'"

smt=ibm\_db.exec\_immediate(dbconn.con,sql.format(id))

res = ibm\_db.fetch\_both(smt) return

render\_template("update.html",update=res)

#delete page code

@app.route("/delete/<string:id>",methods=['POST','GET']) def delete(id):

sql="DELETE FROM authentication WHERE id='{0}'"

smt=ibm\_db.exec\_immediate(dbconn.con,sql.format(id)) return redirect("/welcome") if \_\_name == ' main ': app.run(debug=True)

Output:

