

Assignment - 2

Team ID : PNT2022TMID35342
Vijay Suriya S - 2019103598

Question 1:

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

The screenshot shows the IBM Db2 on Cloud web interface. The 'Tables' tab is selected, displaying a list of tables. The 'USER' table is highlighted, and its definition is shown on the right. The table has four columns: EMAIL, USERNAME, ROLLNO, and PASSWORD. The data types are VARCHAR, VARCHAR, DECIMAL, and VARCHAR respectively. The lengths are 32, 32, 10, and 32. The nullability is N for all columns. The scale is 0 for all columns.

Name	Data type	Nullable	Length	Scale
EMAIL	VARCHAR	N	32	0
USERNAME	VARCHAR	N	32	0
ROLLNO	DECIMAL	N	10	0
PASSWORD	VARCHAR	N	32	0

Question 2:

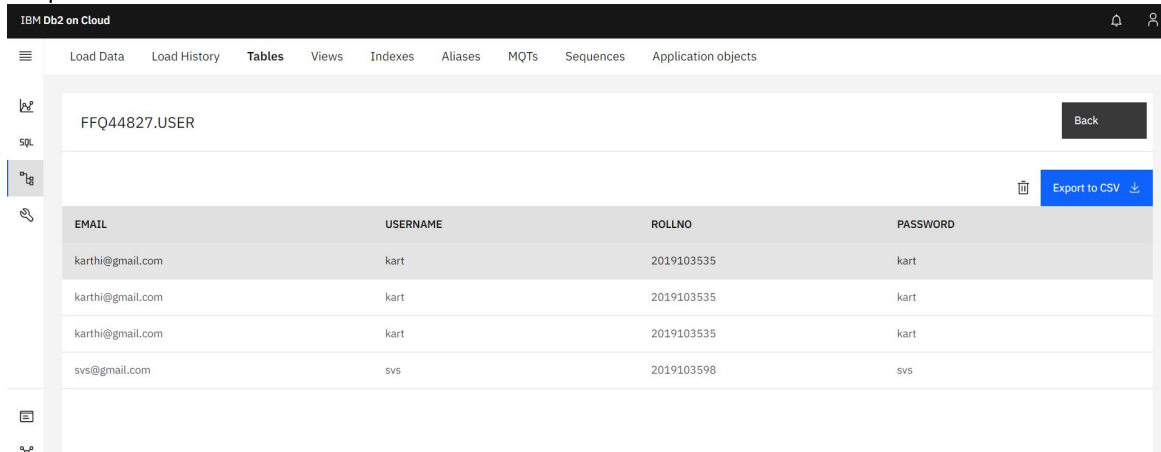
Perform UPDATE, DELETE Queries with user table

```
Terminal Help db.py - IBM-NT - Visual Studio Code

app.py db.py x

Python_DB2 > User Registration > DB2 > db.py > ...
1 import ibm_db
2
3 def list_all(conn):
4     sql = "SELECT * From USER"
5     result = ibm_db.exec_immediate(conn, sql)
6     print(result)
7     dictionary = ibm_db.fetch_both(result)
8     while dictionary!=False:
9         print("Username: "+ dictionary[1])
10        print("Roll Number: "+dictionary[2])
11        dictionary = ibm_db.fetch_both(result)
12
13 def insert_value(conn,email,username,roll,password):
14     sql = "INSERT into USER VALUES('{}','{}',{},{})".format(email,username,roll,password)
15     res = ibm_db.exec_immediate(conn, sql)
16     print("Number of affected rows : ", ibm_db.num_rows(res))
17
18 def delete_value(conn,username):
19     sql = "DELETE From User Where Username = '{}'.format(username)
20     res = ibm_db.exec_immediate(conn,sql)
21     print("Number of affected rows ",ibm_db.num_rows(res))
22
23 try:
24     conn = ibm_db.connect("DATABASE=bludb; HOSTNAME=98538591-7217-4024-b027-8baa776ffad1.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud;
25     PORT=30875; SECURITY = SSL; SSLServerCertificate=DigicertGlobalRootCA.crt; PROTOCOL=TCP;UID=ffq44827;PWD=1FfRv3CjGBVw6mwG;", "", "")
26     # conn is the object to store the connected information
27     print("Connected to Database!! :)")
28     insert_value(conn,"kar@gmail.com","kar", 2019103535, "kar")
29     list_all(conn)
30     delete_value(conn, "kar")
31     insert_value(conn,"karthi@gmail.com","kart", 2019103535, "kart")
32     list_all(conn)
33 except:
34     print(":( db connection failed!")
35     print(ibm_db.conn_errormsg())
```

Output:



IBM Db2 on Cloud

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

FFQ44827.USER

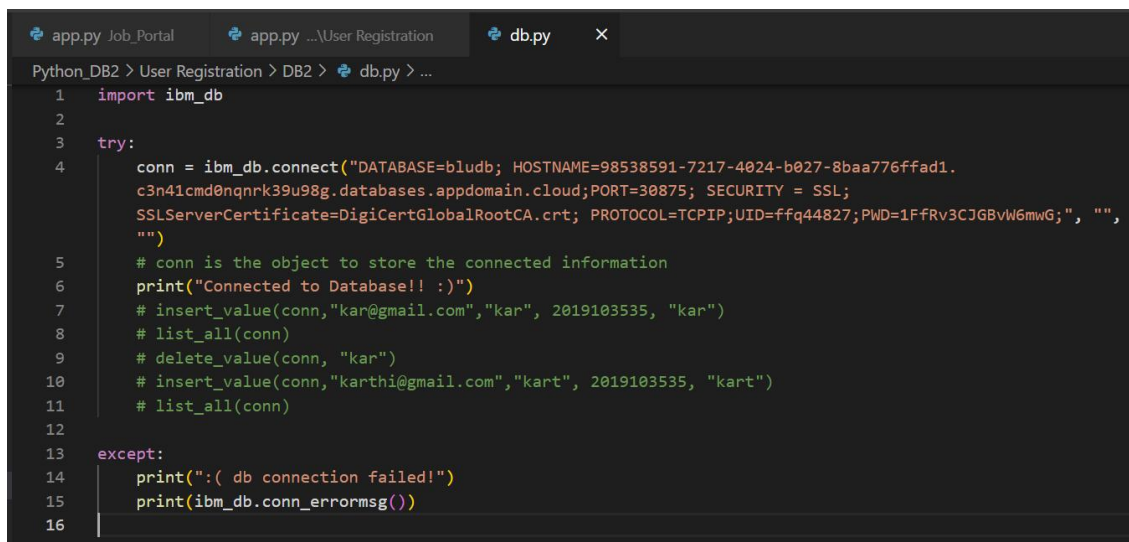
Back

Export to CSV

EMAIL	USERNAME	ROLLNO	PASSWORD
karthi@gmail.com	kart	2019103535	kart
karthi@gmail.com	kart	2019103535	kart
karthi@gmail.com	kart	2019103535	kart
svs@gmail.com	svs	2019103598	svs

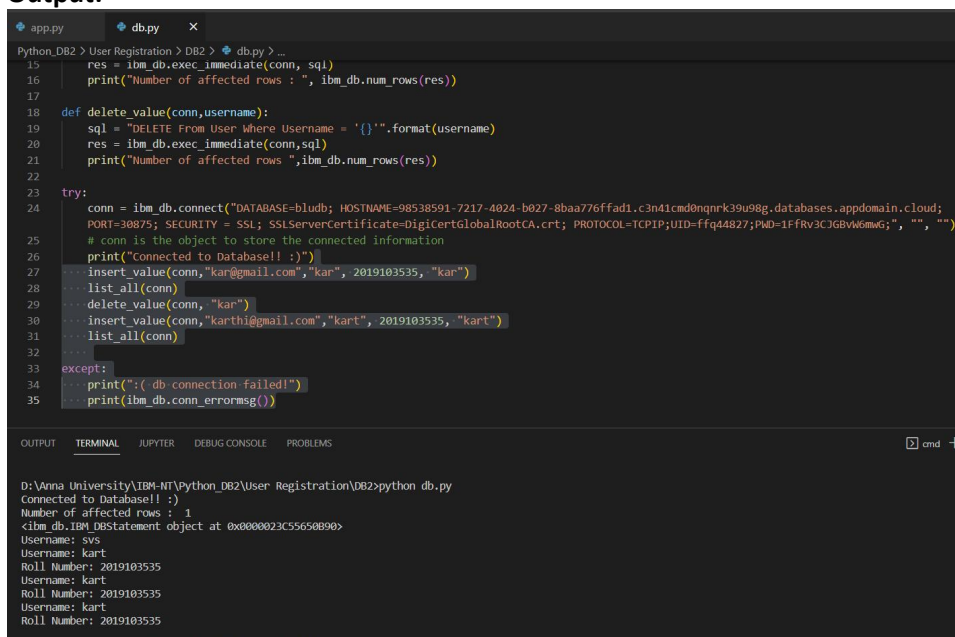
Question 3:

Connect python code to db2.



```
1 import ibm_db
2
3 try:
4     conn = ibm_db.connect("DATABASE=bludb; HOSTNAME=98538591-7217-4024-b027-8baa776ffad1.
        c3n41cmd0nqnk39u98g.databases.appdomain.cloud;PORT=30875; SECURITY = SSL;
        SSLServerCertificate=DigiCertGlobalRootCA.crt; PROTOCOL=TCPIP;UID=ffq44827;PWD=1FfRv3CjGBvW6mwG;", "", "")
5     # conn is the object to store the connected information
6     print("Connected to Database!! :)")
7     # insert_value(conn,"kar@gmail.com","kar", 2019103535, "kar")
8     # list_all(conn)
9     # delete_value(conn, "kar")
10    # insert_value(conn,"karthi@gmail.com","kart", 2019103535, "kart")
11    # list_all(conn)
12
13 except:
14     print(":( db connection failed!")
15     print(ibm_db.conn_errormsg())
16
```

Output:



```
15 res = ibm_db.exec_immediate(conn, sql)
16 print("Number of affected rows : ", ibm_db.num_rows(res))
17
18 def delete_value(conn,username):
19     sql = "DELETE From User Where Username = '{}'".format(username)
20     res = ibm_db.exec_immediate(conn,sql)
21     print("Number of affected rows ",ibm_db.num_rows(res))
22
23 try:
24     conn = ibm_db.connect("DATABASE=bludb; HOSTNAME=98538591-7217-4024-b027-8baa776ffad1.c3n41cmd0nqnk39u98g.databases.appdomain.cloud;
        PORT=30875; SECURITY = SSL; SSLServerCertificate=DigiCertGlobalRootCA.crt; PROTOCOL=TCPIP;UID=ffq44827;PWD=1FfRv3CjGBvW6mwG;", "", "")
25     # conn is the object to store the connected information
26     print("Connected to Database!! :)")
27     insert_value(conn,"kar@gmail.com","kar", 2019103535, "kar")
28     list_all(conn)
29     delete_value(conn, "kar")
30     insert_value(conn,"karthi@gmail.com","kart", 2019103535, "kart")
31     list_all(conn)
32
33 except:
34     print(":( db connection failed!")
35     print(ibm_db.conn_errormsg())
```

OUTPUT TERMINAL JUPYTER DEBUG CONSOLE PROBLEMS

```
D:\Anna University\IBM-NT\Python_Db2\User Registration\082>python db.py
Connected to Database!! :)
Number of affected rows : 1
<ibm_db.IBM Db2Statement object at 0x0000023C55650890>
Username: svs
Username: kart
Roll Number: 2019103535
Username: kart
Roll Number: 2019103535
Username: kart
Roll Number: 2019103535
```

EMAIL	USERNAME	ROLLNO	PASSWORD
karthi@gmail.com	kart	2019103535	kart
karthi@gmail.com	kart	2019103535	kart
karthi@gmail.com	kart	2019103535	kart
svs@gmail.com	svs	2019103598	svs

Question 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 Job_Portal db.py app.py ...\\User Registration X
Python_DB2 > User Registration > app.py > ...
1  from flask import Flask,render_template,request
2  import DB2 as db
3
4  app = Flask(__name__)
5
6
7  @app.route('/',methods=['GET', 'POST'])
8  def login():
9      global userid
10     msg = ''
11     if request.method == 'POST':
12         email = request.form['email']
13         password = request.form['password']
14         sql = "SELECT * FROM USERS WHERE EMAIL=? AND PASSWORD=?"
15         stmt = ibm_db.prepare(conn,sql)
16         ibm_db.bind_param(stmt,1,email)
17         ibm_db.bind_param(stmt,2,password)
18         ibm_db.execute(stmt)
19         res = ibm_db.fetch_assoc(stmt)
20         print(res)
21         if res:
22             session['loggedin'] = True
23             session['id'] = res['USERNAME']
24             userid = res['USERNAME']
25             session['username'] = res['USERNAME']
26             msg = res['USERNAME']
27
28             return render_template('dashboard.html', msg = msg)
29         else:
30             msg = "Incorrect username/password!!"
31             return render_template('login.html', msg = msg)
32
33 @app.route('/register',methods=['GET', 'POST'])
34 def register():
35     msg = ''
36     if request.method == 'POST':
37         username = request.form['username']
```

```
33 @app.route('/register',methods=['GET', 'POST'])
34 def register():
35     msg = ''
36     if request.method == 'POST':
37         username = request.form['username']
38         email = request.form['email']
39         password = request.form['password']
40         sql = "SELECT * FROM USERS WHERE USERNAME=?"
41         stmt = ibm_db.prepare(conn,sql)
42         ibm_db.bind_param(stmt,1,username)
43         ibm_db.execute(stmt)
44         res = ibm_db.fetch_assoc(stmt)
45         print(res)
46         if res:
47             msg = "Account already exists !!"
48         elif not re.match(r'^[a-zA-Z0-9]+@[a-zA-Z0-9]+\.[a-zA-Z]{2,}$',email):
49             msg = "Invalid Email address"
50         elif not re.match(r'^[A-Za-z0-9]+$',username):
51             msg = "Name must contain only characters and numbers !!"
52         else:
53             sql = "INSERT INTO USERS VALUES (?,?)"
54             stmt = ibm_db.prepare(conn,sql)
55             ibm_db.bind_param(stmt,1,username)
56             ibm_db.bind_param(stmt,2,email)
57             ibm_db.bind_param(stmt,3,password)
58             ibm_db.execute(stmt)
59             msg = "Successgully registered !!login to continue"
60             return render_template('login.html', msg = msg)
61         elif request.method == 'POST':
62             msg = "Please fill out the form !"
63             return render_template('register.html', msg = msg)
64
65 if __name__ == '__main__':
66     app.run()
67
```

Output:

localhost:5000/create_user

Gmail Maps Imported IBM project An Easy Way to Ma... GRAPHS PATTERNS Notifications

User Login

Email

Abimanyu@gmail.com

Roll Number

2019103595

Username

karthick

Password

.....

SignUp

existing User? Login

localhost:5000/signup

Gmail Maps Imported IBM project An Easy Way to Ma... GRAPHS PATTERNS Notification

Values Inserted Successfully

Login

localhost:5000

Gmail Maps Imported IBM project An Easy Way to Ma... GRAPHS PATTERNS Notifications

User Login

Username

karthick

Password

.....

Login

new User? Register

Hai you logged in successfully

Cover

Cover your page.

Cover is a one-page template for building simple and beautiful home pages. Download, edit the text, and add your own fullscreen background photo to make