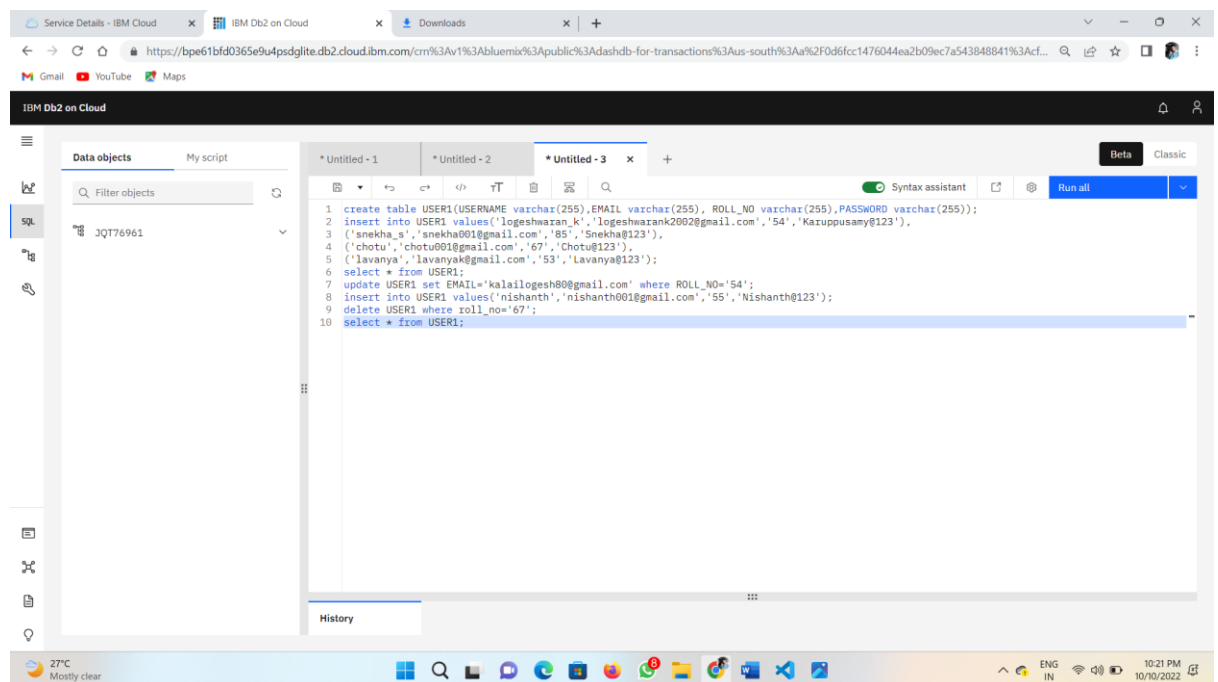


# ASSIGNMENT – II

**1, 2: - Create user table with email USERNAME ,ROLL\_NO, PASSWORD and perform insert ,update and delete.**

## QUERY:

```
create table USER1(USERNAME varchar(255),EMAIL varchar(255), ROLL_NO
varchar(255),PASSWORD varchar(255));
insert into USER1
values('logeshwaran_k','logeshwarank2002@gmail.com','54','Karuppusamy@123'),
('sneka_s','sneka001@gmail.com','85','Sneka@123'),
('chotu','chotu001@gmail.com','67','Chotu@123'),
('lavanya','lavanyak@gmail.com','53','Lavanya@123');
select * from USER1;
update USER1 set EMAIL='kalailogesh80@gmail.com' where ROLL_NO='54';
insert into USER1 values('nishanth','nishanth001@gmail.com','55','Nishanth@123');
delete USER1 where roll_no='67';
select * from USER1;
```



Service Details - IBM Cloud | IBM Db2 on Cloud | Downloads

https://bpe61bfd0365e9u4psdglite.db2.cloud.ibm.com/cm%3Av1%3Abluemix%3Apublic%3Adashdb-for-transactions%3Aus-south%3Aa%2F0d6fcc1476044ea2b09ec7a543848841%3Acf...

IBM Db2 on Cloud

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

JQT76961.USER1

Back

Export to CSV

USERNAME	EMAIL	ROLL_NO	PASSWORD
lavanya	lavanyak@gmail.com	53	Lavanya@123
logeshwaran_k	kalailogesh80@gmail.com	54	Karuppusamy@123
nishanth	nishanth001@gmail.com	55	Nishanth@123
sneha_s	sneha001@gmail.com	85	Sneha@123

27°C Mostly clear

10:22 PM 10/10/2022

### 3. Connect python to db2

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

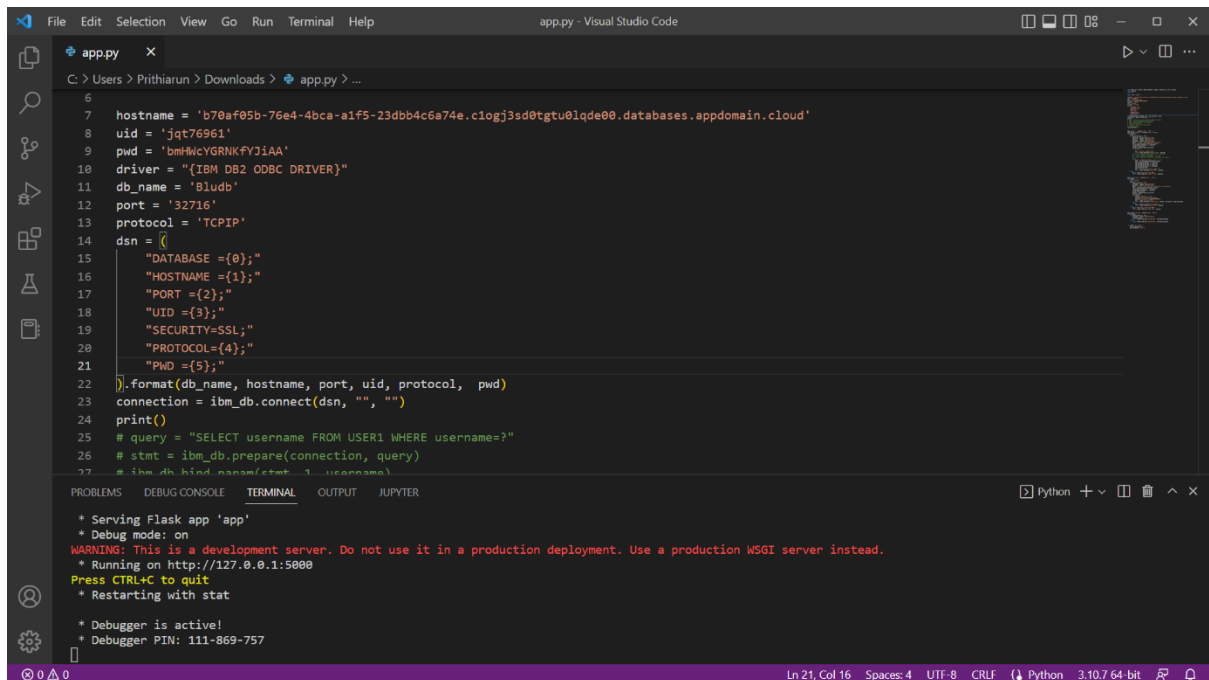
hostname = 'b70af05b-76e4-4bca-a1f5-
23dbb4c6a74e.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud'
uid = 'jqt76961'
pwd = 'bmHWcYGRNKfYJiAA'
driver = "{IBM DB2 ODBC DRIVER}"
db_name = 'Bludb'
port = '32716'
protocol = 'TCPIP'
cert="C:/Users/loges/OneDrive/Desktop/ASSGN_NO_2/certi.crt"
dsn = (
    "DATABASE={0};"
    "HOSTNAME={1};"
    "PORT={2};"
    "UID={3};"
    "SECURITY=SSL;"
    "PROTOCOL={4};"
    "SSLServerCertificate={5};"
    "PWD={6};"
).format(db_name, hostname, port, uid, protocol, cert, pwd)
print(dsn)
try:
    print("Connecting to db2.....")
    db2 = ibm_db.connect(dsn, "", "")
```

```

print()
print("Connected to database")
print("Connection Successful!!!")

except Exception as exception:
    print("unable to connect ", exception)

```



The screenshot shows the Visual Studio Code interface with a file named `app.py` open. The code in the editor defines database connection parameters and attempts to connect to an IBM DB2 database. The terminal output shows the application running successfully on `http://127.0.0.1:5000`.

```

C:\Users\Prithiaron\Downloads> app.py > ...
6
7 hostname = 'b70af05b-76e4-4bca-a1f5-23dbb4c6a74e.clogj3sd0tgtu0lqde00.databases.appdomain.cloud'
8 uid = 'jqt76961'
9 pwd = 'bmHWcYGRNKfYJiAA'
10 driver = "{IBM DB2 ODBC DRIVER}"
11 db_name = 'Bludb'
12 port = '32716'
13 protocol = 'TCPIP'
14 dsn = (
15     "DATABASE={0};"
16     "HOSTNAME={1};"
17     "PORT={2};"
18     "UID={3};"
19     "SECURITY=SSL;"
20     "PROTOCOL={4};"
21     "PWD={5};"
22 ).format(db_name, hostname, port, uid, protocol, pwd)
23 connection = ibm_db.connect(dsn, "", "")
24 print()
25 # query = "SELECT username FROM USER1 WHERE username=?"
26 # stmt = ibm_db.prepare(connection, query)
27 # ibm_db.bind_param(stmt, 1, username)

```

PROBLEMS DEBUG CONSOLE TERMINAL OUTPUT JUPYTER

```

* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 111-869-757

```

Ln 21, Col 16 Spaces: 4 UTF-8 CRLF Python 3.10.7 64-bit

## 4) ACCESS LOGIN WITH CONNTING TO DATABASE

```

from flask import Flask, render_template, request, redirect, url_for, session
import ibm_db
import re

app = Flask(__name__)

hostname = 'b70af05b-76e4-4bca-a1f5-
23dbb4c6a74e.clogj3sd0tgtu0lqde00.databases.appdomain.cloud'
uid = 'jqt76961'
pwd = 'bmHWcYGRNKfYJiAA'
driver = "{IBM DB2 ODBC DRIVER}"
db_name = 'Bludb'
port = '32716'
protocol = 'TCPIP'
cert="C:/Users/loges/OneDrive/Desktop/ASSGN_NO_2/certi.crt"
dsn = (
    "DATABASE={0};"
    "HOSTNAME={1};"
    "PORT={2};"

```

```

"UID ={3};"
"SECURITY=SSL;"
"PROTOCOL={4};"
"SSLServerCertificate={5};"
"PWD ={6};"
).format(db_name, hostname, port, uid, protocol, cert, pwd)
connection = ibm_db.connect(dsn, "", "")
print()
# query = "SELECT username FROM USER1 WHERE username=?"
# stmt = ibm_db.prepare(connection, query)
# ibm_db.bind_param(stmt, 1, username)
# ibm_db.execute(stmt)
# username = ibm_db.fetch_assoc(stmt)
# print(username)
app.secret_key = 'a'

@app.route('/', methods=['GET', 'POST'])
@app.route('/register', methods=['GET', 'POST'])
def register():
    msg = " "
    if request.method == 'POST':
        username = request.form['username']
        email_id = request.form['email_id']
        phone_no = request.form['phone_no']
        password = request.form['password']
        query = "SELECT * FROM USER1 WHERE username=?;"
        stmt = ibm_db.prepare(connection, query)
        ibm_db.bind_param(stmt, 1, username)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        if (account):

            msg = "Account already exists!"
            return render_template('register.html', msg=msg)
        # elif not re.match(r'^@+@[^@]+\.[^@]+', email_id):
        #     msg = "Invalid email address"
        # elif not re.match(r'[A-Za-z0-9+]', username):
        #     msg = "Name must contain only characters and numbers"
        else:
            query = "INSERT INTO USER1 values(?,?,?,?)"
            stmt = ibm_db.prepare(connection, query)
            ibm_db.bind_param(stmt, 1, username)
            ibm_db.bind_param(stmt, 2, email_id)
            ibm_db.bind_param(stmt, 3, phone_no)
            ibm_db.bind_param(stmt, 4, password)
            ibm_db.execute(stmt)
            msg = 'You have successfully Logged In!!'
            return render_template('login.html', msg=msg)

```

```

else:
    msg = 'PLEASE FILL OUT OF THE FORM'
    return render_template('register.html', msg=msg)

@app.route('/login', methods=['GET', 'POST'])
def login():
    global userid
    msg = ''
    if request.method == "POST":
        username = request.form['username']
        password = request.form['password']
        query = "select * from user1 where username=? and password=?"
        stmt = ibm_db.prepare(connection, query)
        ibm_db.bind_param(stmt, 1, username)
        ibm_db.bind_param(stmt, 2, password)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print(account)
        if account:
            session['Loggedin'] = True
            session['id'] = account['USERNAME']
            session['username'] = account['USERNAME']
            msg = 'Logged in Successfully'
            return render_template('welcome.html', msg=msg,
username=str.upper(username))
        else:
            msg = 'Incorrect Username or Password'
            return render_template('login.html', msg=msg)
    else:
        msg = 'PLEASE FILL OUT OF THE FORM'
        return render_template('login.html', msg=msg)

@app.route('/welcome', methods=['GET', 'POST'])
def welcome():
    if request.method == 'POST':
        username = request.form['username']
        print(username)
        return render_template('welcome.html', username=username)
    else:
        return render_template('welcome.html', username=username)

if __name__ == "__main__":
    app.run(debug=True)
    app.run(host='0.0.0.0')

```