

Assignment -2

Python Programming

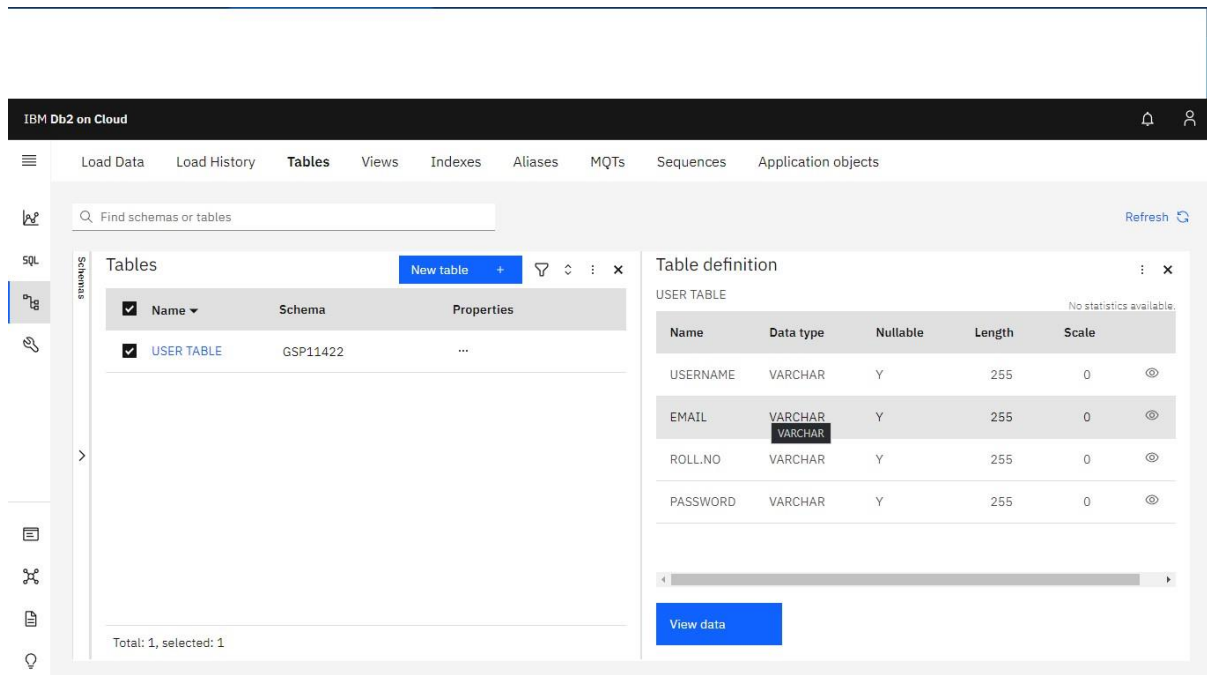
Assignment Date	28 Oct 2022
Student Name	Dhanush M
Team ID	PNT2022TMID31186
Maximum Marks	2 Marks

Question-1:

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

Solution:

Create table students



Question-2:

Perform UPDATE,DELETE queries with user table

Solution:

Insert into user values ("sathish",23,'sarkarsathish@gmail.com','s@thish');

Insert into user values (“shanthini”,25,’shanu@gmail.com’,”shanu@cse”);

Insert into user values (“Narmatha”,20,’narmatha@gmail.com’,”narmu”);

Insert into user values (“vignesh”,37, ’vignesh@gmail.com’,”vicky”);

Alter table user add age int;

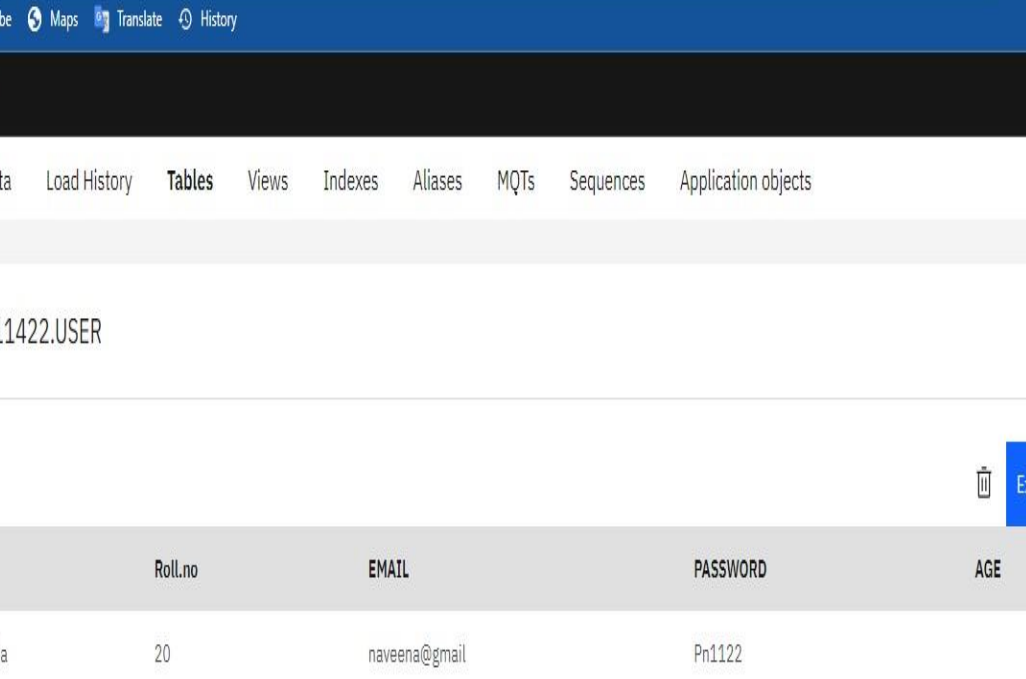
Drop table user;

The screenshot shows the IBM Db2 on Cloud console interface. The top navigation bar includes the IBM logo and several tabs: "CAD-B8-2A4E(Evening Session)", "Service Details - IBM Cloud", and "IBM Db2 on Cloud". The main content area is divided into two sections: "Data objects" on the left and "My script" on the right. The "Data objects" section shows a search bar and a list of objects, including "GSP11422". The "My script" section contains a SQL script editor with the following code:

```
1 insert into user values('sathish',23,'savior@gmail','savior123');
2 insert into user values('narmatha',20,'naveena@gmail','Pn1122');
3 insert into user values('vignesh',42,'badyvicky@gmail','badyvicky45');
4 insert into user values('shanthini',25,'shanuvasu@gmail','shanu25');
5 alter table user add age int;
6
```

Below the script editor, there is a "History" tab showing a table of script execution history. The table has columns for "Script", "Date", "Status", and "Runtime".

Script	Date	Status	Runtime
insert into user values('sathish',23,'savior@gmail','savior123')		❌	0.010 s
insert into user values('narmatha',20,'naveena@gmail','Pn1122')		❌	0.011 s
insert into user values('vignesh',42,'badyvicky@gmail','badyvicky...')		❌	0.009 s
insert into user values('shanthini',25,'shanuvasu@gmail','shanu25...')		❌	0.017 s
Untitled - 2	Oct 5, 2022 11:44:12 AM	✅ 4	0.023 s
insert into user values('sathish',23,'savior@gmail','savior123')		✅	0.005 s



The screenshot shows the IBM Db2 on Cloud console 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 'GSP11422.USER' is displayed. A 'Back' button is in the top right. An 'Export to CSV' button is in the top right of the table area. The table has the following data:

NAME	Roll.no	EMAIL	PASSWORD	AGE
narmatha	20	naveena@gmail	Pn1122	
narmatha	20	naveena@gmail	Pn1122	
sathish	23	savior@gmail	savior123	
sathish	23	savior@gmail	savior123	
shanthini	25	shanuvasu@gmail	shanu25	
shanthini	25	shanuvasu@gmail	shanu25	
vignesh	42	badyvicky@gmail	badyvicky45	

IBM

Service Details - IBM Cloud

IBM Db2 on Cloud

bs2ipcul0apon0jufi80lite.db2.cloud.ibm.com/cm%3Av1%3Abluemix%3Apublic%3Adashdb-for-transactions%3Aeu-gb%3Aa%2F3212639087ac452c95398a907...

GmailYouTubeMapsTranslateHistory

IBM Db2 on Cloud

Data objects

My script

Filter objects

GSP11422

SQL

*Untitled - 2

BetaClassic

Syntax assistant

Run all

1

2

3

4

5

6

insert into user values('sathish',23,'savior@gmail','savior123');

insert into user values('narmatha',20,'naveena@gmail','Pn1122');

insert into user values('vignesh',42,'badyvicky@gmail','badyvicky45');

insert into user values('shanthini',25,'shanuvasu@gmail','shanu25');

History

Results

Find history

Script	Date	Status	Runtime
^ Untitled - 2	Oct 5, 2022 11:44:12 AM	4	0.023 s
insert into user values('sathish',23,'savior@gmail','savior12...			0.005 s
insert into user values('narmatha',20,'naveena@gmail','Pn1122...			0.011 s
insert into user values('vignesh',42,'badyvicky@gmail','badyv...			0.003 s
insert into user values('shanthini',25,'shanuvasu@gmail','sha...			0.004 s

IBM

CAD-B8-244E(Evening Session)-I

Service Details - IBM Cloud

IBM Db2 on Cloud

bs2ipcul0apon0juf80lite.db2.cloud.ibm.com/cm%3Av1%3Abluemix%3Apublic%3Adashdb-for-transactions%3Aeu-gb%3Aa%2F3212639087ac452c95398...

GmailYouTubeMapsTranslateHistory

IBM Db2 on Cloud

HWCADM0002E: Table GSP11422.USER can't be found.

Show logs

X

Filter objects

GSP11422

Run all

```
1 insert into user values('sathish',23,'savior@gmail','savior123');
2 insert into user values('nazmatha',20,'naveena@gmail','Pn1122');
3 insert into user values('vignesh',42,'badyvicky@gmail','badyvicky45');
4 insert into user values('shanthini',25,'shanuvasu@gmail','shanu25');
5 alter table user add age int;
6 drop table user;
```

History

Results

Find history

Script	Date	Status	Runtime
Untitled - 2	Oct 5, 2022 12:06:20 PM	6	0.108 s
insert into user values('sathish',23,'savior@gmail','savior123')			0.016 s
insert into user values('nazmatha',20,'naveena@gmail','Pn1122')			0.021 s
insert into user values('vignesh',42,'badyvicky@gmail','badyvicky...			0.014 s
insert into user values('shanthini',25,'shanuvasu@gmail','shanu25...			0.015 s
alter table user add age int			0.026 s

IBM Db2 on Cloud

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

GSP11422.USER Back

Export to CSV

NAME	Roll.no	EMAIL	PASSWORD
narmatha	20	naveena@gmail	Pn1122
sathish	23	savior@gmail	savior123
shanthini	25	shanuvasu@gmail	shanu25
vignesh	42	badyvicky@gmail	badyvicky45

Question-3:

Connect python code to db2 **Solution:** from flask import Flask,
render_template, request, redirect, url_for, session import ibm_db
import re

```
app = Flask(__name_)
```

```
app.secret_key = 'a' conn
```

```
=
```

```
ibm_db.conect("DATABASE=;HOSTNAME=;PORT=;SECURITY=SSL;SSL  
ServerCertificate=;UID=;PhD=", ' ', ' ')
```

```
@app.route('/') def
```

```
home():
```

```
    return render_template('home.html')
```

```
@app.route('/Login', methods=['GET', 'POST'])
```

```
def login(): global
```

```
    userid msg = ''
```

```
if request.method == 'POST':
```

```
    username      =      request.form['username']      password      =
    request.form['password'] return render_template('home.html') sql =
    "SELECT * FROM Users WHERE username=? AND password=?" stmt
    = ibm_db.prepare(conn, sql) 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']      userid      =
    account['USERNAME'] session['username'] =
    account['USERNAME'] else:
```

```
    msg = 'Incorrect username/password' return
    render_template('login.html',      msg=msg)
    @app.route('/register',      methods=['GET',
    'POST']) def register():
```

```
if request.method == 'POST': username =
    request.form['username'] email = request.form['email']
    password = request.form['password'] sql = "SELECT *
    FROM users WHERE username =?" stmt =
    ibm_db.prepare(conn, sql) ibm_db.bind_param(stmt,
    1, username) ibm_db.execute(stmt) account =
    ibm.db.fetch_assoc(stmt) print(account) if account:
```

```
    msg = "Account already exists!"
```

```
elif not re.match(r'^[a-zA-Z0-9]+@[a-zA-Z0-9]+\.[a-zA-Z0-9]+', email):
```

```
    msg = "format does not match"
```

```
elif not re.match(r'[A-Za-z0-9]+', username):
```

```
    msg = "name must contain characters and numbers" else:
```

```
    insert_sql = "INSERT INTO users VALUES(?, ?, ?)"
```

```
    prep_stmt = ibm_db.prepare(conn.insert_sql)
```

```

        ibm_db.bind_param(prepare_stmt, 1, username)
        ibm_db.bnd_param(prepare_stmt, 2, email)
        ibm_db.bind_param(prepare_stmt, 3, password)
        ibm_db.execute(prepare_stmt) msg = "You have
        successfully registered"

    elif (request.method == "POST"): msg == "Please
        fill out the form" return
        render_template('register.html', msg=msg)
    @app.route('/dashboard') def dash():
    return render_template('dashboard.html')

```

```

@app.route('/apply', methods=['GET',"POST"])
def app(): msg = ' ' if request.method ==
"POST": username = request.form['username']
email = request.form['email'] qualification =
request.form['qualification'] skills =
request.form['skills'] jobs = request.form['s']
stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt, 1, username)
ibm_db.execute(stmt) account =
ibm_db.fetch_assoc(stmt) print(account)

if account():
    msg = "there is only 1 job position" return
    render_template('apply.html', msg=msg)

```

```

insert_sql = "INSERT INTO job VALUES(?, ?, ?, ?, ?)"
prepare_stmt = ibm_db.prepare(conn, insert_sql)
ibm_db.bind_param(prepare_stmt, 1, username)
ibm_db.bind_param(prepare_stmt, 2, email)
ibm_db.bind_param(prepare_stmt, 3, qualification)

```



```

        ibm_db.bind_param(prepare_stmt, 4, skills)
        ibm_db.bind_param(prepare_stmt, 5, jobs)
        ibm_db.execute(prepare_stmt) msg = "You have
        successfully applied for job" session['loggedin'] = True
        TEXT = "Hello user,a new application for job position" + job + isrequested
        "

elif request.method == "POST" msg =
    "Please fill out the form"
    return render_template('register.html', msg=msg)

@app.route('/display')
def display():
    print session["username"], session['id'] cursor =
    mysql.connection.cursor() cursor.execute('SELECT*FROM job
    WHERE userid=%s', (session['id'],)) account = cursor.fetchone()
    print("accountdisplay", account)

```

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 **Solution:**

```

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

```

```

app = Flask(__name__)

```

```

app.secret_key = 'a'

```

conn

=

```
ibm_db.conect("DATABASE=bludb;HOSTNAME=21fecfd8-47b7-4937-840d-d791d0218660.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;PORT=31864;SECURITY=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;USERNAME=gsp11422;PASSWORD=ixh307unZIzgLLB9", '', '')
```

```
@app.route('/') def
```

```
home():
```

```
    return render_template('home.html')
```

```
@app.route('/Login', methods=['GET', 'POST'])
```

```
def login(): global userid msg = ''
```

```
if request.method == 'POST':
```

```
    username = request.form['username'] password =
```

```
    request.form['password'] sql = "SELECT * FROM
```

```
    Users WHERE username=?
```

```
AND password=?" stmt =
```

```
    ibm_db.prepare(conn, sql)
```

```
    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']          userid          =
    account['USERNAME'] session['username'] =
    account['USERNAME'] msg='Logged in
    successfully!'
else:
    msg = 'Incorrect username/password' return
    render_template('login.html',msg=msg)

```

```

@app.route('/register', methods=['GET', 'POST']) def
register():
    if request.method == 'POST': username =
    request.form['username'] email = request.form['email']
    password = request.form['password'] sql = "SELECT *
    FROM users WHERE username =?" stmt =
    ibm_db.prepare(conn, sql) ibm_db.bind_param(stmt,
    1, username) ibm_db.execute(stmt)
    account =
    ibm_db.fetch_assoc(stmt)
    print(account) if account:
        msg = "Account already exists!"
    elif not re.match(r'^@]+@[^@]+\.[^@]+', email):

```

```

        msg = "Invalid email address"
    elif not re.match(r'[A-Za-z0-9+', username):
        msg = "name must contain characters and numbers" else:
        insert_sql = "INSERT INTO users VALUES(?, ?,
        ?)" prep_stmt = ibm_db.prepare(conn.insert_sql)
        ibm_db.bind_param(prepare_stmt, 1, username)
        ibm_db.bnd_param(prepare_stmt, 2, email)
        ibm_db.bind_param(prepare_stmt, 3, password)
        ibm_db.execute(prepare_stmt) msg = "You have
        successfully registered"
    elif request.method == "POST":
        msg == "Please fill out the form"
    return render_template('register.html', msg=msg)

```

```

@app.route('/dashboard')
def dash():
    return render_template('dashboard.html')

```

```

@app.route('/apply', methods=['GET',"POST"])
def app(): msg = ' ' if request.method ==
"POST": username = request.form['username']
email = request.form['email'] qualification =
request.form['qualification'] skills =
request.form['skills'] jobs = request.form['s']

```

```
stmt          =          ibm_db.prepare(conn,sql)
ibm_db.bind_param(stmt,      1,      username)
ibm_db.execute(stmt)          account          =
ibm_db.fetch_assoc(stmt) print(account)
```

```
if account():
```

```
    msg = "there is only 1 job position" return
    render_template('apply.html',      msg=msg)
    insert_sql = "INSERT INTO job VALUES(?,
    ?, ?, ?, ?)" prep_stmt = ibm_db.prepare(conn,
    insert_sql) ibm_db.bind_param(prepare_stmt, 1,
    username) ibm_db.bind_param(prepare_stmt, 2,
    email) ibm_db.bind_param(prepare_stmt, 3,
    qualification) ibm_db.bind_param(prepare_stmt,
    4, skills) ibm_db.bind_param(prepare_stmt, 5,
    jobs) ibm_db.execute(prepare_stmt) msg = "You
    have successfully applie for job"
    session['Loggedin'] = True
```

```
    TEXT = "Hello user,a new application for job position + job
+ is requested"
```

```
elif request.method == "POST": msg
    = "Please fill out the form"
    return render_template('register.html', msg=msg)
```

```

@app.route('/display') def
display():
    print
    session["username"], session['id'] cursor =
    mysql.connection.cursor()
    cursor.execute('SELECT*FROM job WHERE userid=%s',
(session['id'],)) account =
    cursor.fetchone()
    print("accountdisplay", account)
return render_template('display.html',account=account)

```

```

@app.route('/logout')

```

```

def logout():
    session.pop('loggedin',None)
    session.pop('id',None)
    session.pop('username',None)
    return render_template('home.html')

```

```

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

```