

## Assignment -2

### IBM CLOUD DB2

Assignment Date	29 September 2022
Student Name	NISHANTH R
Student Roll Number	73151921033
Maximum Marks	2 Marks

### Questions:

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

The screenshot displays the IBM Db2 on Cloud web interface. The top navigation bar includes 'Load Data', 'Load History', 'Tables', 'Views', 'Indexes', 'Aliases', 'MQTs', 'Sequences', and 'Application objects'. The 'Tables' tab is active, showing a list of tables in the 'KWR62447' schema. A table named 'USERS' is selected. The 'Table definition' panel on the right shows the structure of the 'USERS' table:

Name	Data type	Nullable	Length	Scale
USERNAME	CHAR	Y	20	0
EMAIL	CHAR	Y	200	0
PASSWORD	CHAR	Y	200	0

At the bottom of the 'Table definition' panel, there is a 'View data' button. The left sidebar shows the 'Schemas' list with 'KWR62447' selected. The bottom status bar indicates 'Total: 1, selected: 0'.

## 2. Perform UPDATE,DELETE Queries with user table.

### UPDATE TABLE:-

The screenshot displays the IBM Db2 on Cloud web interface. The top navigation bar includes links for Service Details, IBM Cloud, and various browser tabs. The main interface is divided into a left sidebar with navigation icons and a central workspace.

**SQL Execution View:**

- Data objects:** A sidebar on the left shows a tree view with 'YDR22992' expanded, containing 'Tables', 'Views', 'MQTs', 'Aliases', and 'Nicknames'. 'USERTABLE' is selected under 'Tables'.
- Script Editor:** The main area shows a script titled '\*Untitled - 1' with the following SQL code:

```
1 INSERT
2 INTO "YDR22992"."USERTABLE" ("NAME", "ROLLNUMBER", "EMAIL", "PASSWORD")
3 VALUES(
4 'sasi', --NAME CHAR(10)
5 46, --ROLLNUMBER INTEGER
6 'sasi@gmail.com', --EMAIL CHAR(20)
7 '1234' --PASSWORD VARCHAR(32)
8 );
9
```
- History:** Below the script editor, a 'History' tab shows a list of executed scripts. The first entry is successful, while the subsequent two failed due to syntax errors.

**Table View:**

- The 'Tables' tab is selected in the top navigation bar.
- The table 'YDR22992.USERTABLE' is displayed with a 'Back' button in the top right.
- An 'Export to CSV' button is located in the top right corner of the table view.
- The table contains the following data:

NAME	ROLLNUMBER	EMAIL	PASSWORD
ramesh	42	ramesh@gmail.com	5678
sasi	46	sasi@gmail.com	1234
sasi	46	sasi@gmail.com	1234

## DELETE TABLE:-

The screenshot displays the IBM Db2 on Cloud web interface. The top navigation bar includes links for Service Details, IBM Db2 on Cloud, and various database tools. The main workspace is divided into three panes: Data objects, My script, and History.

**Data objects pane:** Shows a tree view of the database structure. The 'Tables' folder is expanded, showing a table named 'YDR22992'. Below it, the 'USERTABLE' is listed under the 'Views' folder.

**My script pane:** Contains a SQL script for deleting the 'USERTABLE' view. The script is as follows:

```
1 DELETE FROM "YDR22992"."USERTABLE"  
2 --Search condition (e.g. WHERE "NAME" = NULL)  
3 WHERE ROLLNUMBER = 46;  
4
```

**History pane:** Displays a table of executed scripts. The table has columns: Script, Date, Status, and Runtime.

Script	Date	Status	Runtime
Untitled - 3	Oct 27, 2022 10:31:37 AM	1	0.005 s
DELETE FROM "YDR22992"."USERTABLE" --Search condition (e.g. WHERE "NAME" = ...		1	0.005 s
Untitled - 2	Oct 27, 2022 10:27:43 AM	1	0.007 s
INSERT INTO "YDR22992"."USERTABLE" ("NAME", "ROLLNUMBER", "EMAIL", "PASSWORD")...		1	0.007 s
Untitled - 2	Oct 27, 2022 10:25:17 AM	1	0.007 s

**Table view:** The 'Tables' tab is selected, showing the 'YDR22992.USERTABLE' table. The table has columns: NAME, ROLLNUMBER, EMAIL, and PASSWORD. The data is as follows:

NAME	ROLLNUMBER	EMAIL	PASSWORD
ramesh	42	ramesh@gmail.com	5678

3. Connect python code to db2.

**NOTE:- Question 4 contains Question 3 answer**

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.

```

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.connect(
    "DATABASE=bludb;HOSTNAME=55fbc997-9266-4331-afd3-888b05e734c0.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;
    PORT=31929;USERNAME=kwr62447;PASSWORD=sl6om9GM9MwckIT;SECURITY=SSL;SSLSERVERCERTIFICATE=DigiCertGlobalRootCA.crt;", "", "")

@app.route("/", methods=['GET', 'POST'])
def register():
    msg = ''
    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 only 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.bind_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('/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 !'

        msg = 'Logged in successfully !'
        return render_template('dashboard.html', msg=msg)
    else:
        msg = 'Incorrect username / password !'
    return render_template('login.html', msg=msg)

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

```

## Register.html

```

<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Smart Fashion Register</title>

    <!-- bootstrap css cdn -->
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
        integrity="sha384-JcKb8q3iqJ61gNV9KGb8thSsNjpSL0n8PARn9HuZOnIxN0hoP+VmmDGMN5t9UJ0Z" crossorigin="anonymous">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.css">
    <!-- css stylesheet --> <a href="/static/style.css">Follow link (ctrl + click)</a>
    <link rel="stylesheet" href="/static/style.css">
    <!-- font styles cdn -->
    <link rel="preconnect" href="https://fonts.gstatic.com">
    <link href="https://fonts.googleapis.com/css2?family=Alegreya&display=swap" rel="stylesheet">
    <link href="https://fonts.googleapis.com/css2?family=Alegreya:wght@600&display=swap" rel="stylesheet">
</head>

<body>
    <!-- bootstrap navbar -->
    <!-- navbar ends -->
    <!-- Login form -->
    <div class="login text-center mt-5">

        <form action="/" method="post">
            <h2 class="form-text-h2"> Register Form </h2>
            <div class="msg">{{ msg }}</div>

```

```

        <input type="text" class="form-control" name="username" placeholder="Enter Your Username" id="username"
        required>
        <input type="email" class="form-control" name="email" placeholder="Enter Your Email ID" id="email" required>
        <input type="password" class="form-control" name="password" placeholder="Enter Your Password" id="password"
        required>
        <button type="submit" id="button" class="btn btn-primary"> Register </button>
        <div class="note mt-3 text-center">
            <!--Register form -->
            <p> already have an account ? please login <a href="/login">login! </a> </p>
        </div>
    </form>
</div>

</body>

</html>

```

## Login.html

```

<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Home page</title>
    <meta charset="UTF-8">

    <!-- bootstrap css cdn -->
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
    integrity="sha384-JcKb8q3iqJ61gNV9KGb8thSsNjpSL0n8PARn9HuZOnIxN0hoP+VmmDGMN5t9UJ0Z" crossorigin="anonymous">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.css">
    <!-- css stylesheet -->
    <link rel="stylesheet" href="/static/style.css">
    <!-- font styles cdn -->
    <link rel="preconnect" href="https://fonts.gstatic.com">
    <link href="https://fonts.googleapis.com/css2?family=Alegreya&display=swap" rel="stylesheet">
    <link href="https://fonts.googleapis.com/css2?family=Alegreya:wght@600&display=swap" rel="stylesheet">
</head>

<body>
    <H1>WELCOME TO HOME PAGE</H1>
    
</body>

</html>

```

## Style.css

```

* {
  margin: 0;
  padding: 0;
}

body {
  background: linear-gradient(to bottom right, #529aec, #d13d1b);
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
}

form {
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  background-color: white;
  width: 40%;
  height: 450px;
  border-radius: 50px;
  margin: 100px;
  gap: 25px;
}

h2 {
  font-size: large;
  font-weight: bolder;
}

input {
  width: 80% !important;
}

div {
  display: flex;
  align-items: center;
  justify-content: center;
  flex-direction: column;
  width: 90%;
}

.form-text-vk {
  font-size: small;
  padding-bottom: 10px;
}

.form-text-h2 {
  font-size: x-large;
  padding-top: 10px;
}

```

Output:-

registration page:-

The image displays two screenshots of a web application interface, likely for a smart fashion system, showing the registration and login pages.

**Top Screenshot: Register Form**

The browser address bar shows the URL: `127.0.0.1:5000`. The page title is "Smart Fashion Register". The registration form is centered on a light blue background. It contains the following fields and elements:

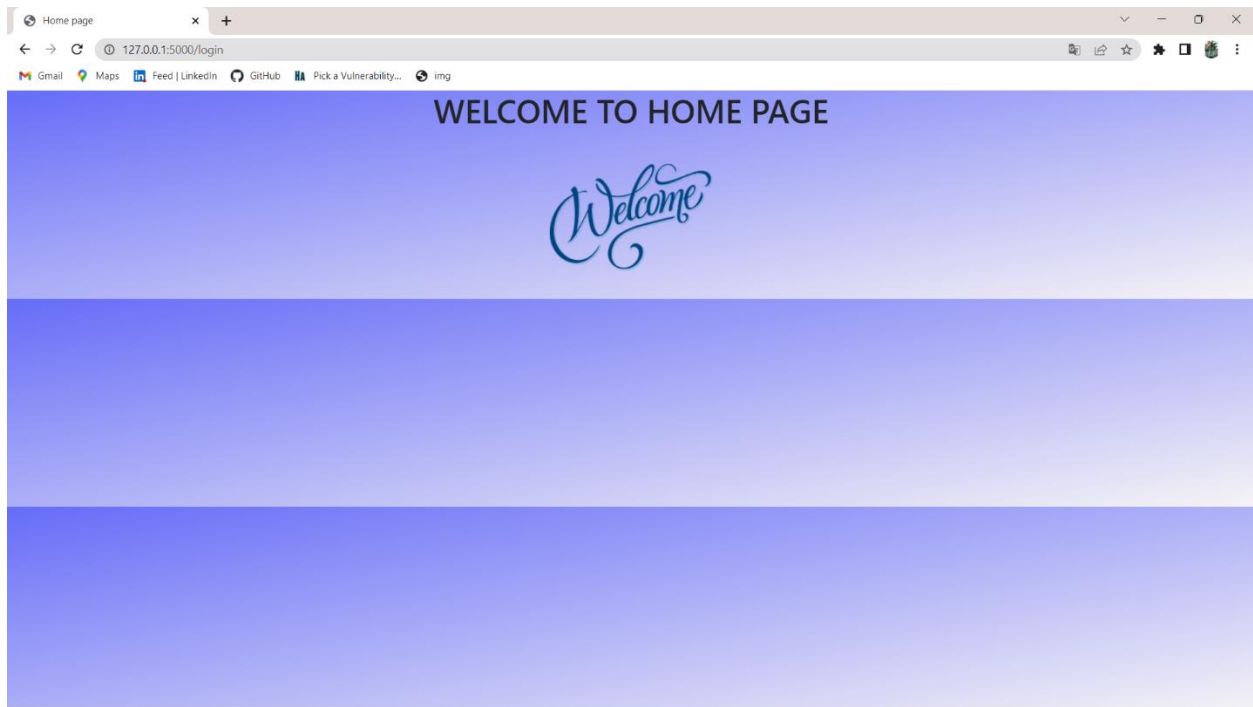
- Register Form** (Section Header)
- Username** field: `nishanth`
- Email** field: `rn05062000@gmail.com`
- Password** field: `*****`
- Register** button
- Link: `already have an account ? please login login!`

**Bottom Screenshot: Login Form**

The browser address bar shows the URL: `127.0.0.1:5000/login`. The page title is "Smart Fashion Login". The login form is centered on a light blue background. It contains the following fields and elements:

- Login Form** (Section Header)
- Username** field: `nishanth`
- Password** field: `*****`
- Login** button
- Link: `Don't have an account yet? Click here to register!`





## DataBase Table:

USERNAME	EMAIL	PASSWORD
ramesh	rameshkrishna002@gmail.com	12345
soundar	sakthisoundarraj26@gmail.com	12345678