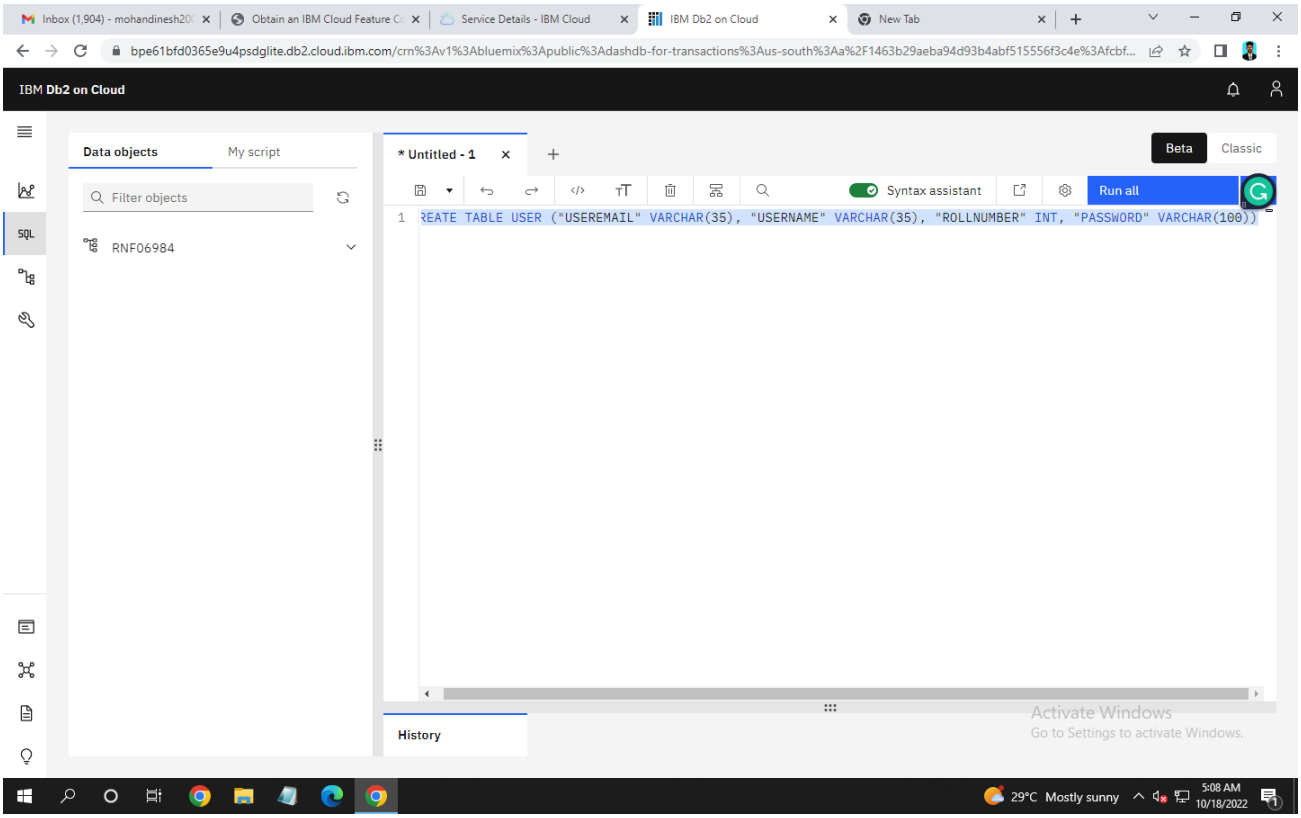
Assignment -2

Connect with Database Assignment

Assignment Date	09 October 2022
Student Name	Kavya P
Student Roll Number	732919CSR050
Maximum Marks	2 Marks

Question-1:

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



2. Perform UPDATE, DELETE Queries with User table

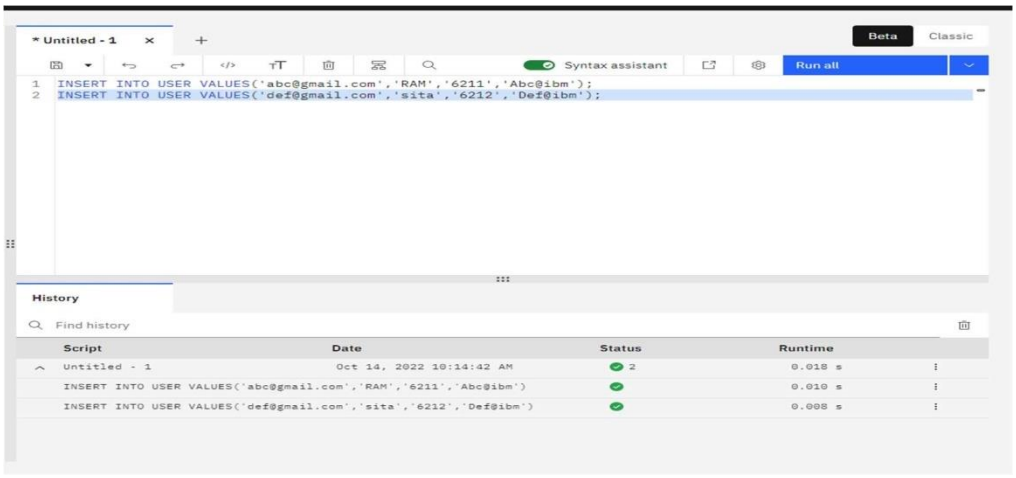
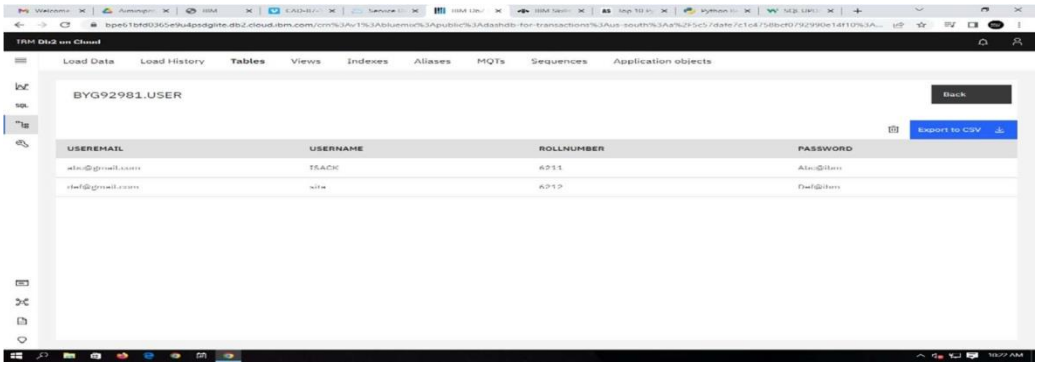
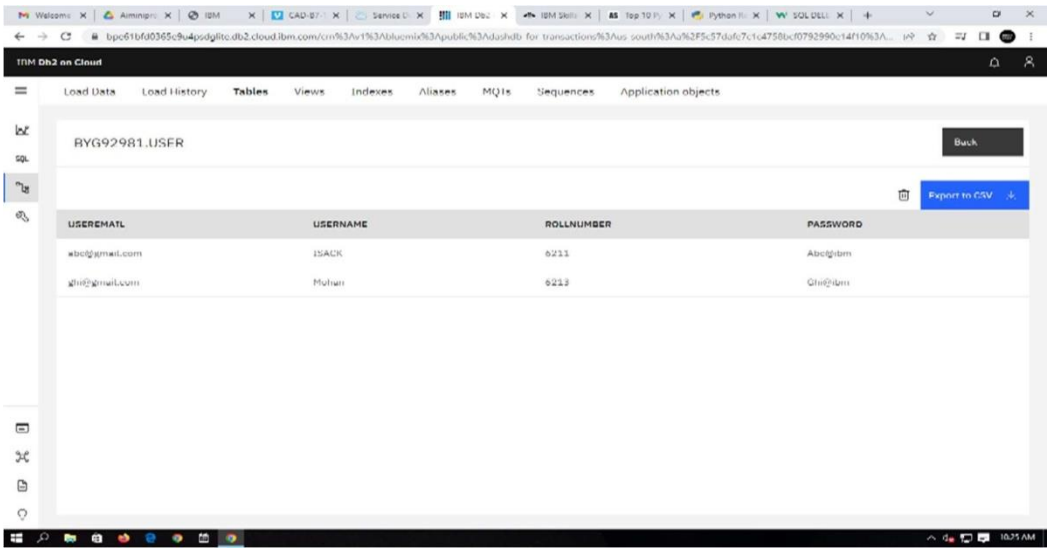


Table View:



UPDATE:

Table View:



DELETE:

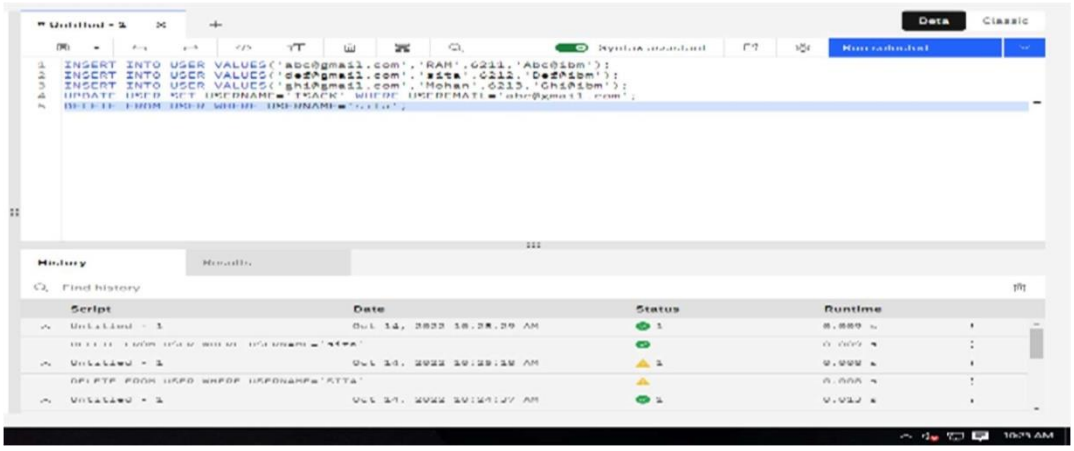
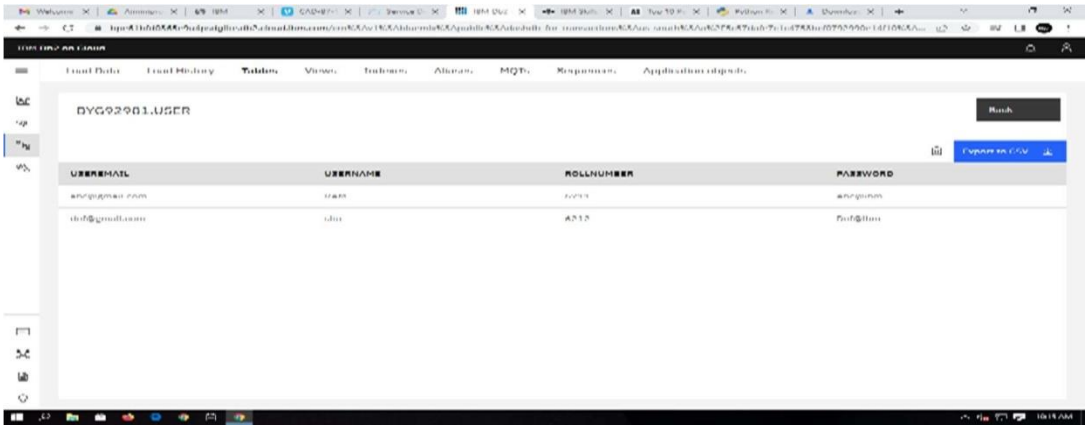


TABLE View:



3. Connect python with db2.

Solution:

```
import ibm_db conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=6667d8e9-9d4d-4ccb-ba32-21da3bb5aafc.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=30376;SECURITY=SSL;SSLS erverCertificate=DigiCertGl obalRootCA.crt;PROTOCOL=TCPIP;UID= rnf06984
;PWD=” VWqiPBgxELVtAn32
","","")
```

4. Create a flask app with the registration page. Login page and the 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 so the welcome page.

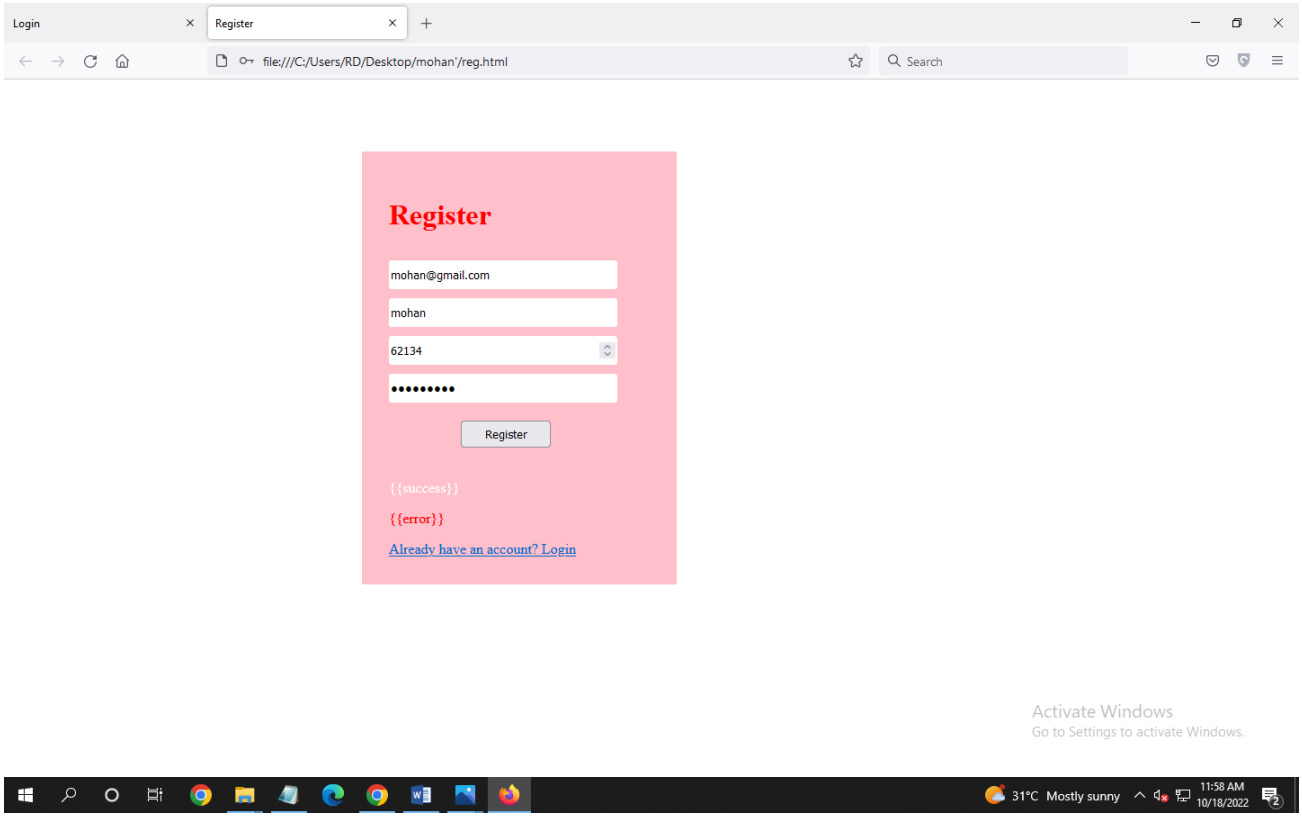
Solution:

app.py

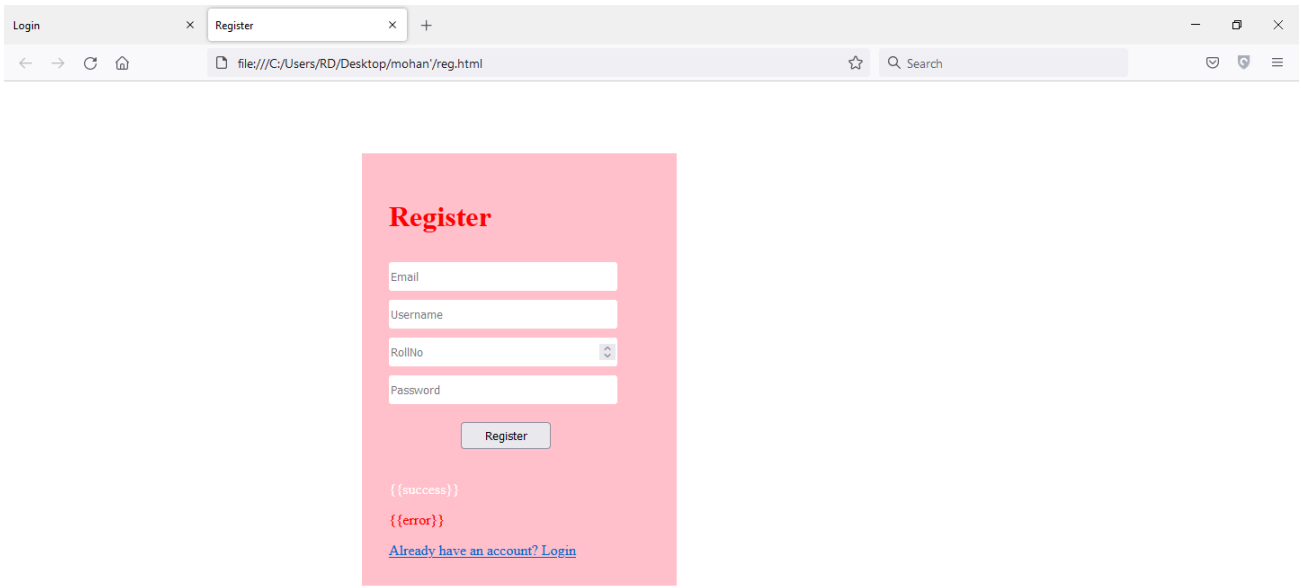
```
from flask import Flask, render_template, request, redirect, url_for,
session

import ibm_db
import bcrypt conn
=
ibm_db.connect("DATABASE=bludb;HOSTNAME=;PORT=;SECURITY=SSL;SSLServerCertific

ate=DigiCer tGlobalRootCA.crt;UID=;PWD=","") # url_for('static', filename='style.css')

app = Flask(_name_)
app.secret_key = 'C21FGSBAPOK43K5VSIDFB2'

@app.route("/",methods=['GET'])
def home():
   if 'email' not in session:
     return redirect(url_for('login'))
   return render_template('home.html',name='Home')

@app.route("/register",methods=['GET','POST'])
def register(): if request.method ==
 'POST': email = request.form['email']
 username = request.form['username']
 rollNo = request.form['rollNo']
   password = request.form['password']

   if not email or not username or not rollNo or not password:
     return render_template('register.html',error='Please fill all fields')

   hash=bcrypt.hashpw(password.encode('utf-8'),bcrypt.gensalt())

   query = "SELECT * FROM USER WHERE email=? OR
   rollNo=?" stmt = ibm_db.prepare(conn, query)
   ibm_db.bind_param(stmt,1,email)
   ibm_db.bind_param(stmt,2,rollNo) ibm_db.execute(stmt)
   isUser = ibm_db.fetch_assoc(stmt)

   if not isUser:
     insert_sql = "INSERT INTO User(username,email,PASSWORD,rollNo) VALUES
     (?,?,?,?)" prep_stmt = ibm_db.prepare(conn, insert_sql)
     ibm_db.bind_param(prep_stmt, 1, username) ibm_db.bind_param(prep_stmt, 2,
     email) ibm_db.bind_param(prep_stmt, 3, hash) ibm_db.bind_param(prep_stmt, 4,
     rollNo) ibm_db.execute(prep_stmt)
     return render_template('register.html',success="You can login")
```

```
    else:        return        render_template('register.html',error='Invalid
  Credentials') return render_template('register.html',name='Home')

@app.route("/login",methods=['GET','POST'])
def login():
    if request.method == 'POST':
      email = request.form['email']
      password = request.form['password']

      if not email or not password:
        return render_template('login.html',error='Please fill all fields')
      query = "SELECT * FROM USER WHERE
      email=?" stmt = ibm_db.prepare(conn, query)
      ibm_db.bind_param(stmt,1,email)
      ibm_db.execute(stmt) isUser =
      ibm_db.fetch_assoc(stmt)
      print(isUser,password)
      if not isUser:
        return render_template('login.html',error='Invalid Credentials')

      isPasswordMatch = bcrypt.checkpw(password.encode('utf-
      8'),isUser['PASSWORD'].encode('utf-
8'))

      if not isPasswordMatch:
        return render_template('login.html',error='Invalid Credentials')

      session['email'] = isUser['EMAIL']
      return redirect(url_for('home'))

    return render_template('login.html',name='Home')
@app.route('/logout') def
logout():
    session.pop('email', None)
    return redirect(url_for('login'))
```

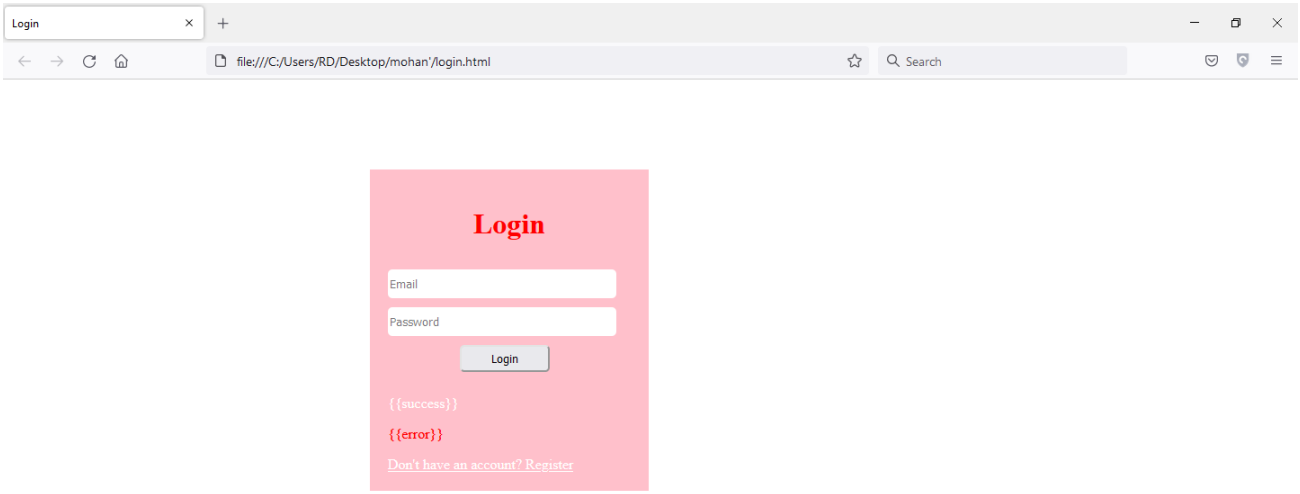
OUTPUT:



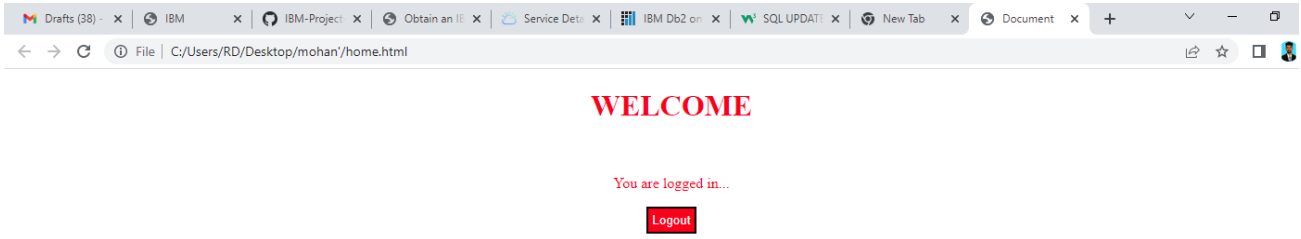
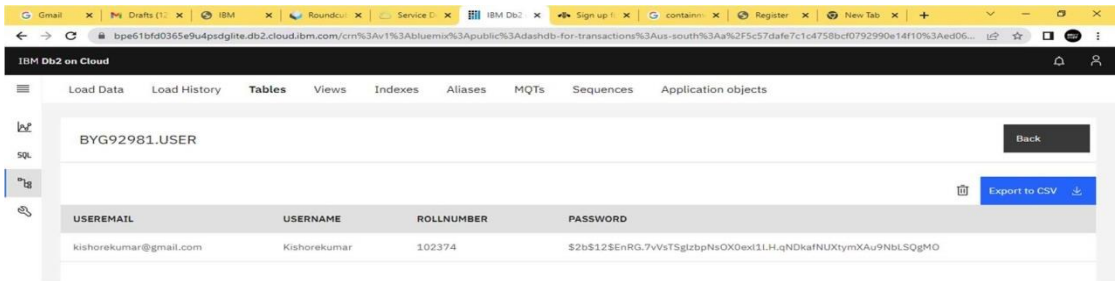
Activate Windows
Go to Settings to activate Windows.



Activate Windows
Go to Settings to activate Windows.



Activate Windows
Go to Settings to activate Windows.



Database: