**Assignment -2**

Python Programming

|  |  |
| --- | --- |
| Assignment Date | 24 September 2022 |
| Student Name | Sobeka T |
| Student Roll Number | 7179KCTKCTKCTKCTKCTKCTKCT19BEC037 |
| Maximum Marks | 2 Marks |

**Question-1:**

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

|  |
| --- |
| **Solution:** |
|  |
|  |
|  | CREATE TABLE User(  RollNo int,  LastName varchar(255),  FirstName varchar(255),  Password varchar(255),  Email varchar(255)  ); |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Question-2:**

2.Perform UPDATE,DELETE Queries with user table.

**Solution:**

CREATE TABLE User(

RollNo int,

LastName varchar(255),

FirstName varchar(255),

Password varchar(255),

Email varchar(255)

);

INSERT INTO User ( RollNo ,LastName ,FirstName, Password,Email)

VALUES (1, 'T', 'Sobeka', 'Sobeka',’xyz@gmail.com’)

INSERT INTO User ( RollNo ,LastName ,FirstName, Password, Email)

VALUES (2, ‘VS’,’Yugesh’,’yugesh@2002’)

INSERT INTO User ( RollNo ,LastName ,FirstName, Password, Email)

VALUES (3, 'Swift', Taylor', '123345', ’xyz@gmail.com’)

INSERT INTO User ( RollNo ,LastName ,FirstName, Password, Email)

VALUES (4,'Gomez', 'Selena','448265’, ’xyz@gmail.com’)

INSERT INTO User ( RollNo ,LastName ,FirstName, Password, Email)

VALUES (5, 'Salmaan. 'Da', 'sijsk5', ’xyz@gmail.com’)

INSERT INTO User ( RollNo ,LastName ,FirstName, Password, Email)

VALUES (6,’ Jenner', 'Kendall', 'hsgyasj5', ’xyz@gmail.com’)

SELECT \* FROM User

UPDATE User

SET LastName=’j’

WHERE LastName=’ Jenner';

UPDATE User

SET FirstName=’Tay’

WHERE FirstName=’ Taylor';

DELETE FROM User

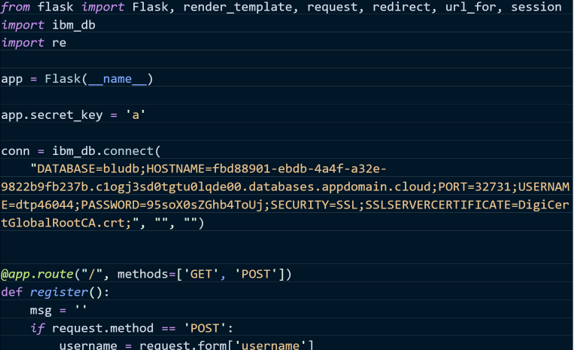
WHERE RollNo=5;

DELETE FROM User

WHERE RollNo=6;

**Question-3:**

3.Connect python code to db2.



**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, url\_for, redirect  from flask\_sqlalchemy import SQLAlchemy  from flask\_login import UserMixin, login\_user, LoginManager, login\_required, logout\_user, current\_user  from flask\_wtf import FlaskForm  from wtforms import StringField, PasswordField, SubmitField  from wtforms.validators import InputRequired, Length, ValidationError  from flask\_bcrypt import Bcrypt  app = Flask(\_\_name\_\_)  db = SQLAlchemy(app)  bcrypt = Bcrypt(app)  app.config['SQLALCHEMY\_DATABASE\_URI'] = 'sqlite:///database.db'  app.config['SECRET\_KEY'] = 'thisisasecretkey'  login\_manager = LoginManager()  login\_manager.init\_app(app)  login\_manager.login\_view = 'login'  @login\_manager.user\_loader  def load\_user(user\_id):  return User.query.get(int(user\_id))  class User(db.Model, UserMixin):  id = db.Column(db.Integer, primary\_key=True)  username = db.Column(db.String(20), nullable=False, unique=True)  password = db.Column(db.String(80), nullable=False)  class RegisterForm(FlaskForm):  username = StringField(validators=[  InputRequired(), Length(min=4, max=20)], render\_kw={"placeholder": "Username"})  password = PasswordField(validators=[  InputRequired(), Length(min=8, max=20)], render\_kw={"placeholder": "Password"})  submit = SubmitField('Register')  def validate\_username(self, username):  existing\_user\_username = User.query.filter\_by(  username=username.data).first()  if existing\_user\_username:  raise ValidationError(  'That username already exists. Please choose a different one.')  class LoginForm(FlaskForm):  username = StringField(validators=[  InputRequired(), Length(min=4, max=20)], render\_kw={"placeholder": "Username"})  password = PasswordField(validators=[  InputRequired(), Length(min=8, max=20)], render\_kw={"placeholder": "Password"})  submit = SubmitField('Login')  @app.route('/')  def home():  return render\_template('home.html')  @app.route('/login', methods=['GET', 'POST'])  def login():  form = LoginForm()  if form.validate\_on\_submit():  user = User.query.filter\_by(username=form.username.data).first()  if user:  if bcrypt.check\_password\_hash(user.password, form.password.data):  login\_user(user)  return redirect(url\_for('dashboard'))  return render\_template('login.html', form=form)  @app.route('/dashboard', methods=['GET', 'POST'])  @login\_required  def dashboard():  return render\_template('dashboard.html')  @app.route('/logout', methods=['GET', 'POST'])  @login\_required  def logout():  logout\_user()  return redirect(url\_for('login'))  @ app.route('/register', methods=['GET', 'POST'])  def register():  form = RegisterForm()  if form.validate\_on\_submit():  hashed\_password = bcrypt.generate\_password\_hash(form.password.data)  new\_user = User(username=form.username.data, password=hashed\_password)  db.session.add(new\_user)  db.session.commit()  return redirect(url\_for('login'))  return render\_template('register.html', form=form)  if \_\_name\_\_ == "\_\_main\_\_":  app.run(debug=True) |
|  |
|  |

