KLE TECH.

KLE Technological University

Creating Value
Leveraging Knowledge

Department of MCA

NAME : Sadhana Umarani

USN : 01FE22BCA237

Div : D

Guide : Sarala D

STRUCTURE OF PRESENTATION

1. Python Basics
2. SQL
3. Django

PYTHON INTRODUCTION

What is Python?

Python is a high-level, interpreted, and general-purpose programming language known for its simplicity and readability Created by **Guido van Rossum** and first released in **1991**, Python has become one of the most popular languages in the world.

Key Features of Python

- Easy to Learn and Use**:** Simple syntax that resembles the English language.
- Interpreted Language**:** Executes code line by line, making debugging easier.

PYTHON INTRODUCTION

- Cross-Platform: Runs on various platforms like Windows, macOS, Linux, and more.
- Open Source: Freely available and supported by a large community.
- Extensive Libraries: Rich collection of libraries for tasks like data analysis, machine learning, web development, and automation.

PYTHON VARIABLES

- Variables are used to store data that can be used later in a program.
- They are created when a value is assigned to them.

Rules for Variable Naming:

- Must begin with a letter or underscore.
- Cannot start with a number.
- Can only contain alphanumeric characters and underscores.
- Case-sensitive (e.g., Name and name are different).

PYTHON DATATYPES

Data types define the kind of data that a variable can hold.

Common Data Types:

- Numeric**:** Integer (int), Float (float)
- Text: String (str)
- Boolean**:** True or False
- Sequence: List, Tuple, Range
- Mapping: Dictionary (dict)

Python is flexible in data types, and variables can change types during execution.

PYTHON OPERATORS

Operators are special symbols that perform operations on variables and values.

Types of Operators:

- Arithmetic Operators: +, -, *, /, %, ** (exponentiation), // (floor division)
- Comparison Operators: ==, !=, >, <, >=, <=
- Logical Operators: and, or, not
- Assignment Operators: =, +=, -=, *=, /=, %=
- Bitwise Operators: &, |, ^, ~, <<, >>

PYTHON LOOPING

Loops are used to repeat a block of code multiple times.

Types of Loops:

- for loop: Iterates over a sequence like a list, tuple, or string.
- while loop: Repeats as long as a condition is true.

Loops help automate repetitive tasks and reduce code redundancy.

Loop Control Statements:

- break: Exits the loop immediately.
- continue: Skips the current iteration and moves to the next.

PYTHON FUNCTIONS

- File handling allows reading, writing, and manipulating files.
- Files can be of different types: text files (.txt), binary files (.bin), etc.
- **File Modes:**
  a. **'r' (Read):** Opens a file for reading (default).
  b. **'w' (Write):** Opens a file for writing (creates a new file if not exists).
  c. **'a' (Append):** Opens a file for adding content at the end.
  d. **'x' (Create):** Creates a new file (error if the file exists).
  e. **'t' (Text mode):** Default mode for handling text files.
  f. **'b' (Binary mode):** For binary files like images and videos.

SQL INTRODUCTION

What is SQL?

- SQL (Structured Query Language) is a standard language used to manage and manipulate databases.
- It allows users to create, read, update, and delete data stored in relational databases.

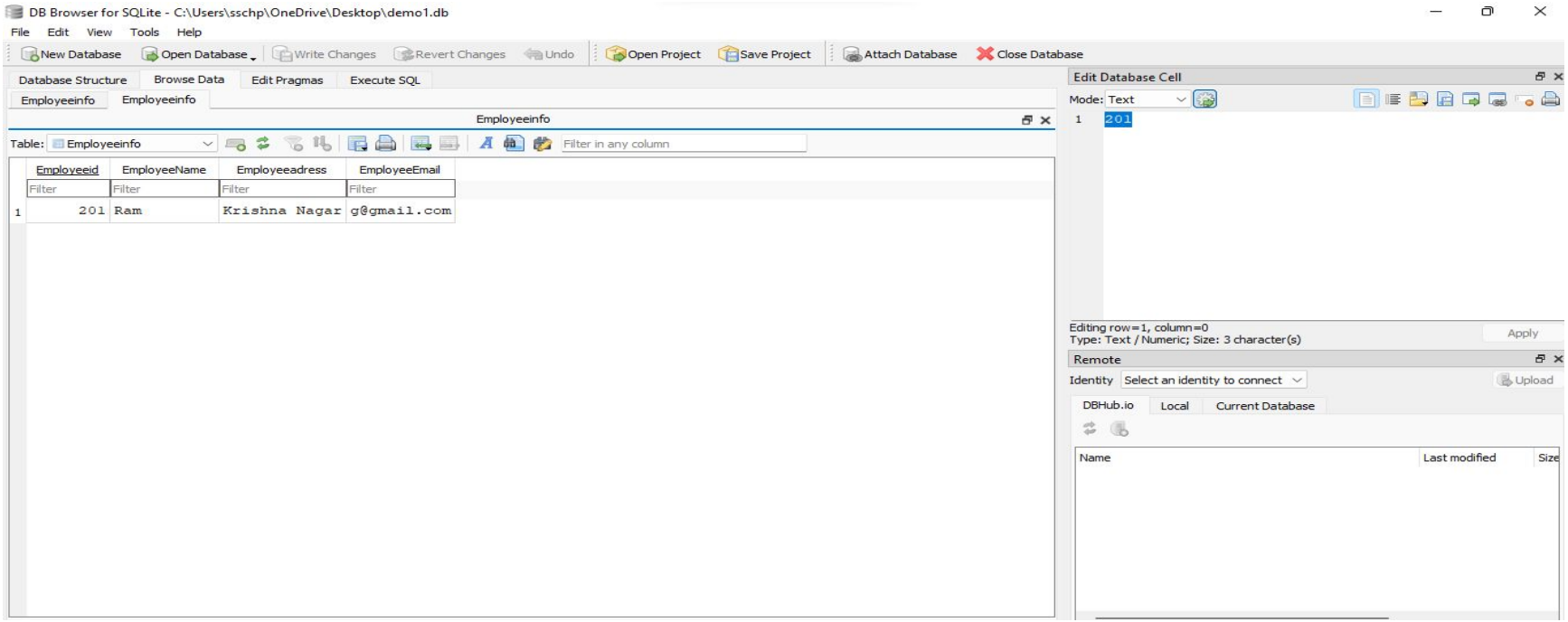
Key Features:

- Simple and easy to learn syntax.
- Used for managing structured data.
- Databases: DB Browser (SQlite)

SQL INTRODUCTION

- Creating Tables: SQL enables users to define the structure of a table, specifying columns, data types, and constraints like primary keys.
- Inserting Data: The language allows adding new records into tables, ensuring that information is stored systematically.
- Retrieving Data: With SQL, users can extract specific data using queries that filter and sort information according to their needs.

SQL INTRODUCTION



DJANGO INTRODUCTION

What is Django?

Django is a high-level, open-source web framework written in **Python** that enables the rapid development of secure and maintainable websites. It was created by **Adrian Holovaty** and **Simon Willison** and first released in **2005**.

Key Features:

- **Fast Development:** Built for speed, reducing development time.
- **Security:** Built-in protection against common web threats like SQL injection and cross-site scripting (XSS).
- **Scalability:** Suitable for projects of all sizes, from small applications to large-scale systems.

CODE 1

```python
from django.http import HttpResponse
from django.template import loader
from .models import MemberInfo
from django.contrib.auth.models import User

def hello(request):
    return HttpResponse("Hello World!")

def hello1(request):
    return HttpResponse("Good morning everyone")

def page1(request):
    template = loader.get_template('first.html')
    return HttpResponse(template.render())

def page2(request):
    return render(request,'second.html')

def members_data(request):
    mymembers=MemberInfo.objects.all().values()
    template = loader.get_template('member_data.html')
    context={
        'mymembers': mymembers,
    }
    return HttpResponse(template.render(context,request))

def datapush(request):
    if request.method == 'POST':
        firstname = request.POST.get('firstname')
        lastname = request.POST.get('lastname')
        x=MemberInfo.objects.create(firstname=firstname,lastname=lastname)
        return redirect('members_data')

    return render(request,'datapush.html')

def member_detail(request,member_id):
    members=MemberInfo.objects.filter(id=member_id)
    if members.exists():
        member=members[0]
    else:
        return HttpResponse("member  not found", status=404)

    template=loader.get_template('member_detail.html')
    context={'member':member}
    return HttpResponse(template.render(context,request))

def register(request):
 if request.method == 'POST':
    username = request.POST['username']
    firstname = request.POST['firstname']
    lastname = request.POST['lastname']
    password = request.POST['password']
    email = request.POST['email']
    user = User.objects.create_user(first_name=firstname,last_name=lastname,username=username, email=email, password=password)
    return redirect('members_data')
 else:
    return render(request, 'register.html')
```

CODE 2

- settings.py | datapush.html | first.html | models.py

test1 > members > templates > first.html

```
<h1> haegl academy </h1>
<p> page1 </p>
<a href="/page2/"> visit Second </a>
```

- settings.py | datapush.html | first.html | models.py

test1 > members > templates > second.html

```
<h1> haegl academy </h1>
<p> page2 </p>
```

CODE 3

Run ··· ← → 🔍 django

settings.py | datapush.html | first.html | models.py | second.html | urls.py | 0001_initia

test1 > members > urls.py > ...

```python
from django.urls import path
from.import views

urlpatterns =[

    path('hello/',views.hello,name='hello'),
    path('hello1/',views.hello1,name='hello1'),
    path('page1/',views.page1,name='page1'),
    path('page2/',views.page2,name='page2'),

    path('members_data/',views.members_data,name='members_data'),
    path('datapush/',views.datapush,name='datapush'),
    path('member_detail/<int:member_id>/',views.member_detail,name='member_detail'),
    path('register/',views.register,name='register'),



]
```

OUTPUT : 1

Members List

preksha bafna> sameksha bafna> lord krishna> rahul bafna> sameksha bafna> sadhana um>

Member details

ID:6

First Name:sadhana

Last Name:um

OUTPUT : 2

haegl academy

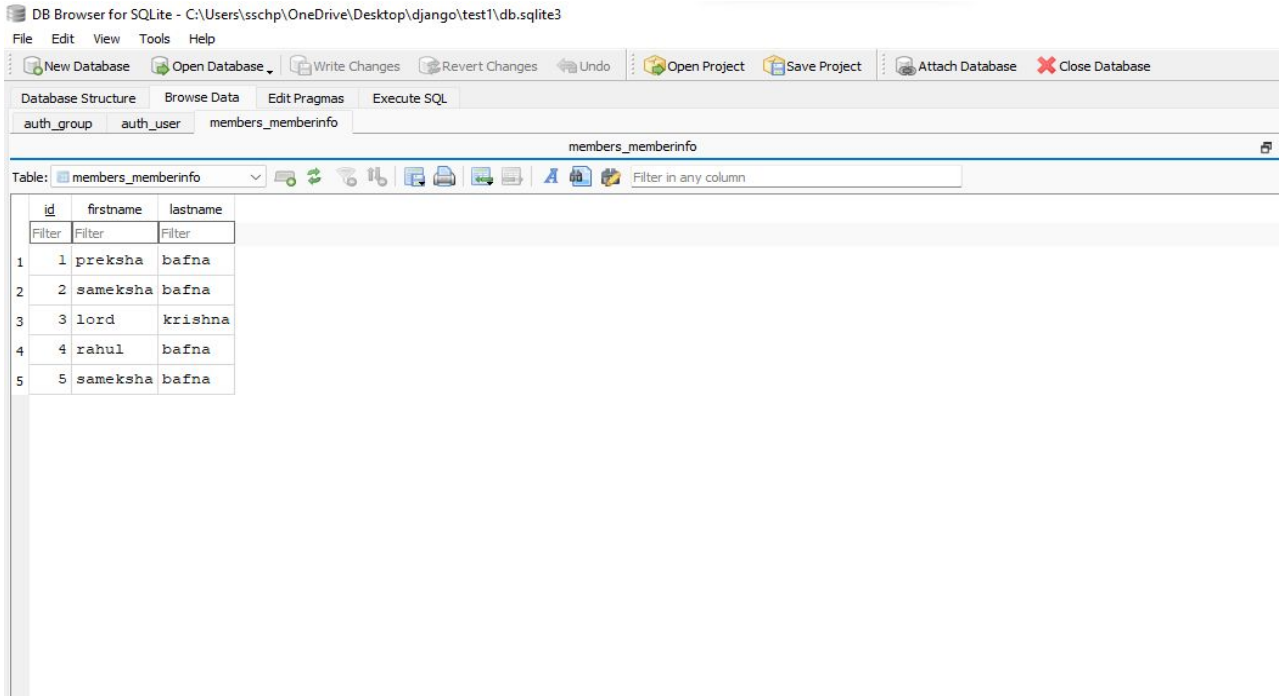
page1

visit Second

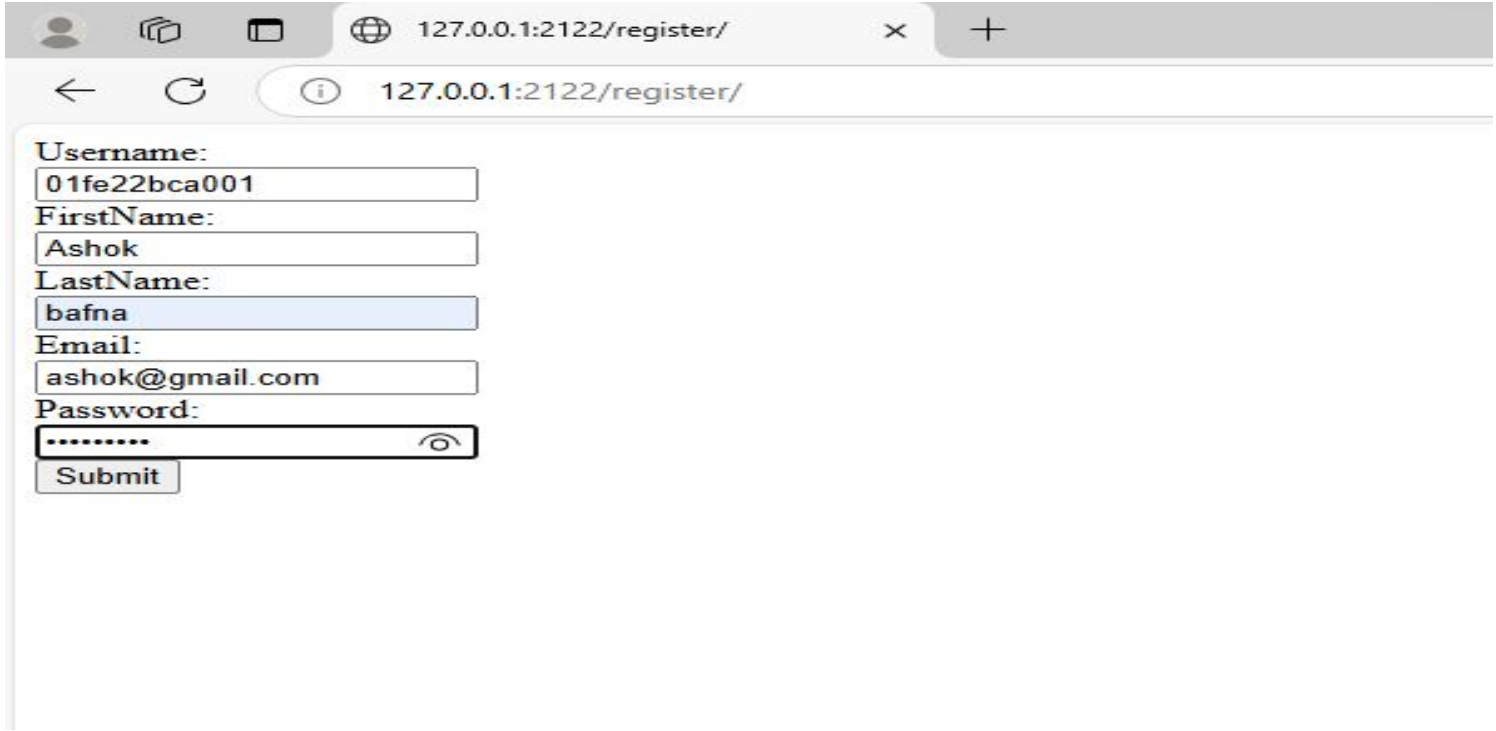
haegl academy

page2

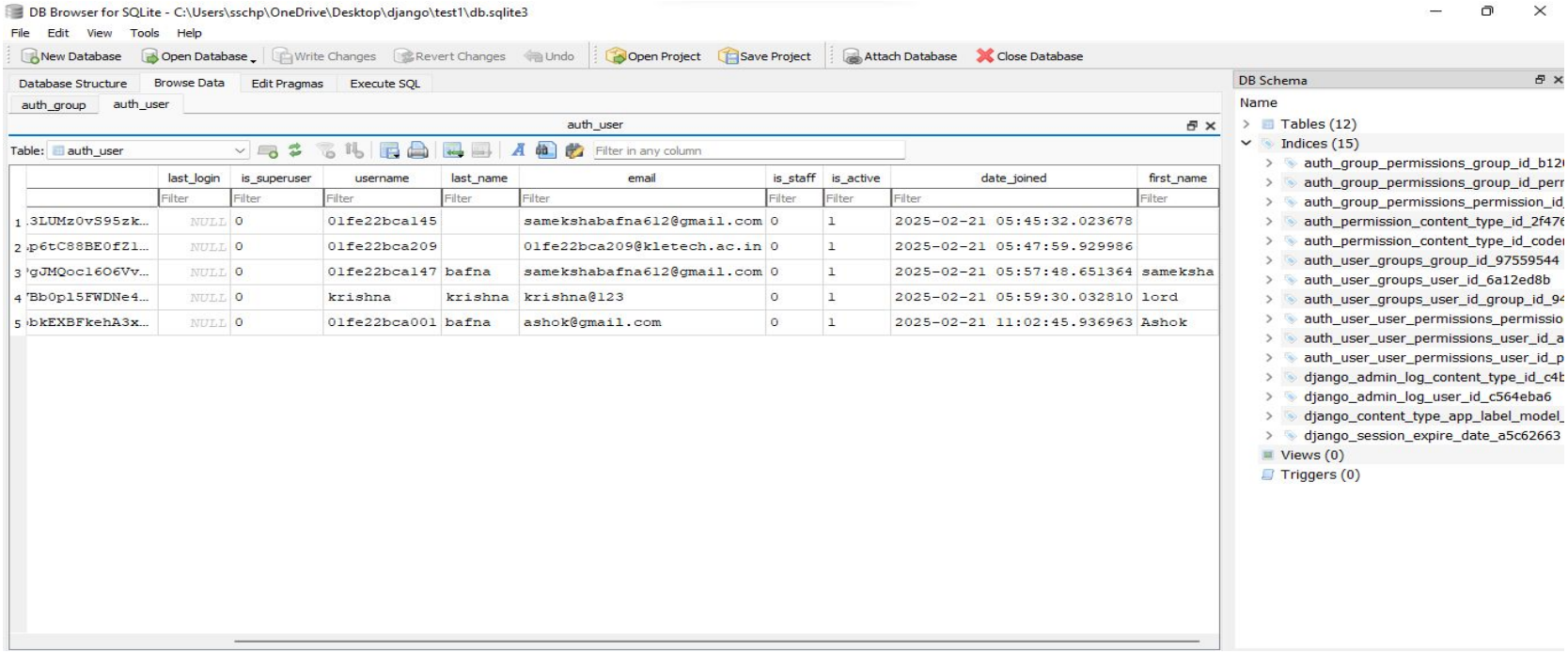
DB BROWSE SQLITE



OUTPUT : 3


127.0.0.1:2122/register/
127.0.0.1:2122/register/
Username:
01fe22bca001
FirstName:
Ashok
LastName:
bafna
Email:
ashok@gmail.com
Password:
Submit

DB BROWSER SQLITE



		last_login	is_superuser	username	last_name	email	is_staff	is_active	date_joined	first_name
		Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	3LUMz0vS95zk...	NULL	0	01fe22bca145		samekshabafna612@gmail.com	0	1	2025-02-21 05:45:32.023678	
2	p6tC88BE0fZ1...	NULL	0	01fe22bca209		01fe22bca209@kletech.ac.in	0	1	2025-02-21 05:47:59.929986	
3	gJMQoc16O6Vv...	NULL	0	01fe22bca147	bafna	samekshabafna612@gmail.com	0	1	2025-02-21 05:57:48.651364	sameksha
4	Bb0p15FWDNe4...	NULL	0	krishna	krishna	krishna@123	0	1	2025-02-21 05:59:30.032810	lord
5	bkEXBFkehA3x...	NULL	0	01fe22bca001	bafna	ashok@gmail.com	0	1	2025-02-21 11:02:45.936963	Ashok

THANK YOU!

HARDWARE REQUIREMENTS

1. RAM: Minimum 8GB
2. Storage: At least 200GB
3. Processor: Intel i5 or higher
4. GPU: Recommended for deep learning model training (NVIDIA GTX 1060 or higher)

SOFTWARE REQUIREMENTS

1. Operating System: Windows/Linux/macOS
2. Backend: Flask/FastAPI (Python)
3. Frontend: React.js, Mapbox, Leaflet.js, Material UI
4. Database: DB BROWSER SQLITE