Plenty of Fish in the S(ea)QL

Cade Lueker, Isabella Robert Llorens, & Bradley Kolar

CSDS 341: *Introduction to database systems*

CASE SCHOOL OF ENGINEERING
CASE WESTERN RESERVE UNIVERSITY

Technologies

For this project we chose to use **Flask, python, & sqlite3**

- Flask
  - simplistic backend and rendering of *jinja html templates*
  - great for apps with only a few pages
  - used *bootstrap css* to provide a better UI
- Python
  - very clean and concise code
  - good for interacting with databases, contains sqlite3 module
  - *Jupyter notebook* allowed for testing parts of the project
- Sqlite3
  - local db so no need for server, perfect for prototyping
  - integrates with python nicely

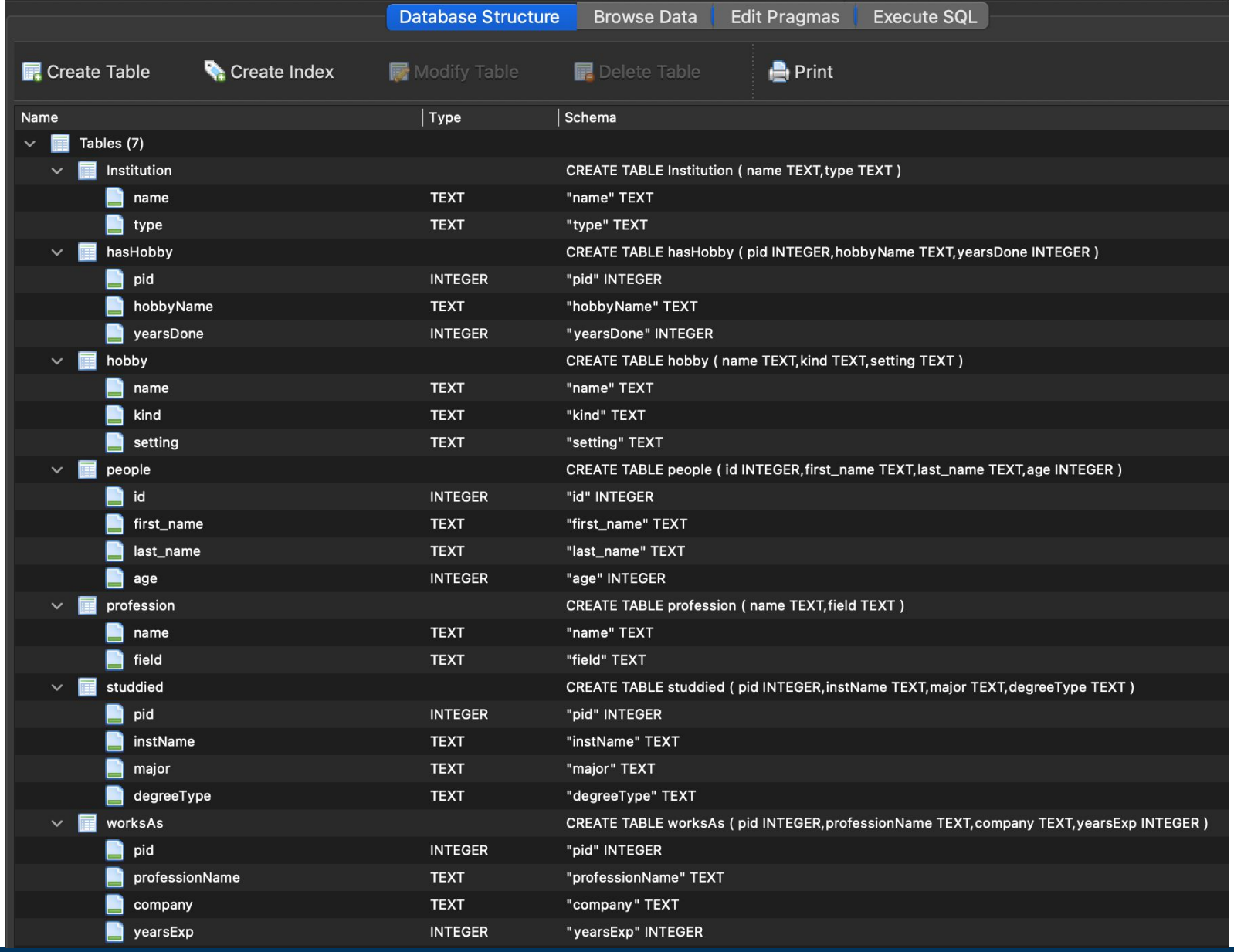
Goals

Despite being a group of 3 we wanted to create a *web UI* and allow for *user input*. There is a python backend that makes a

- **Database** with…
  - people
  - hobbies
  - professions
  - institutions (schools)
  - and their corresponding relations
    - worksAs, studied, hasHobby

and a python/ flask / jinja

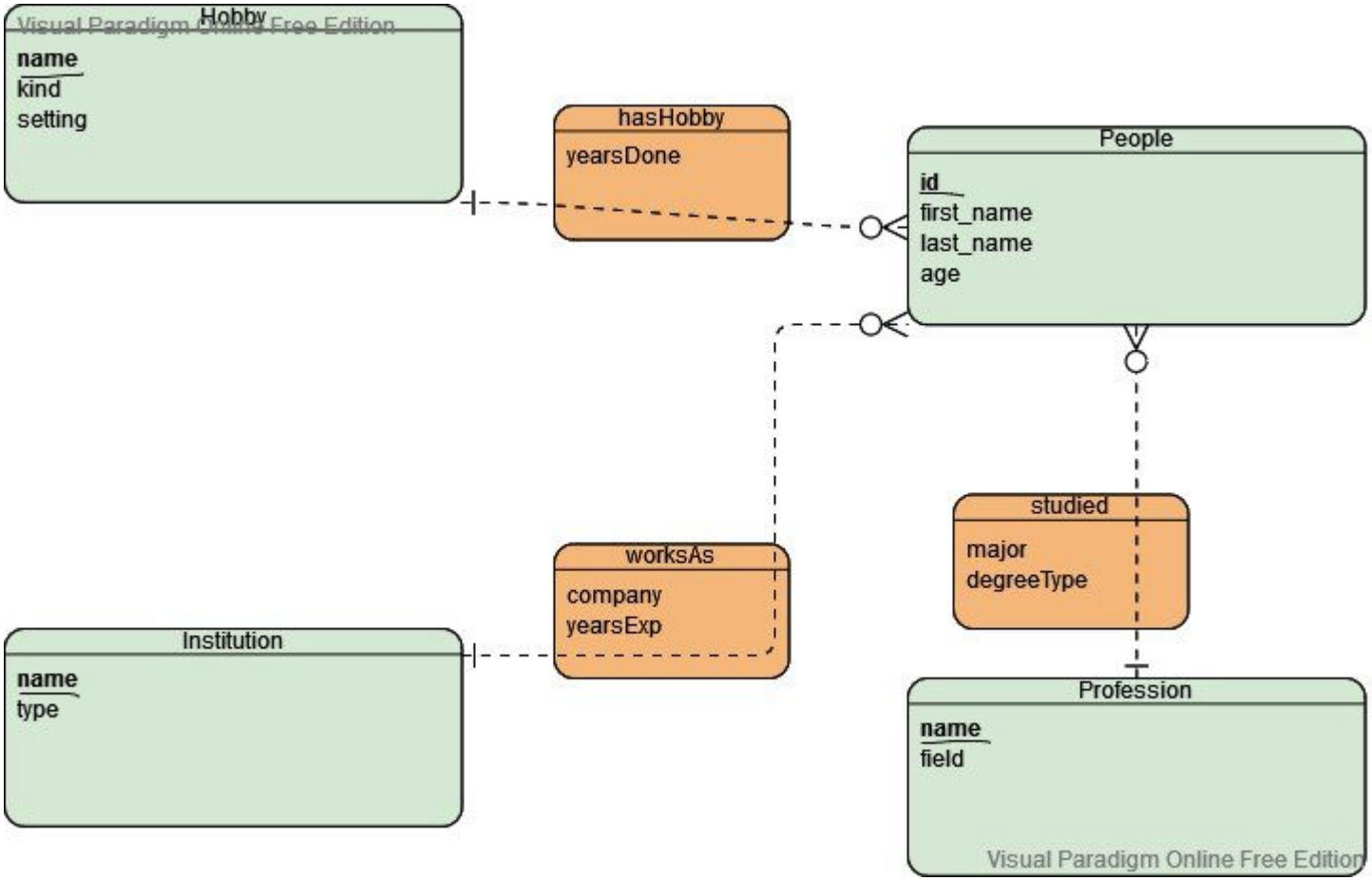
- a **front end** that…
  - takes user preferences in a form
  - returns the people from the database that meet *all* criterion provided



Name	Type	Schema
Tables (7)		
Institution		CREATE TABLE Institution (name TEXT,type TEXT)
name	TEXT	"name" TEXT
type	TEXT	"type" TEXT
hasHobby		CREATE TABLE hasHobby (pid INTEGER,hobbyName TEXT,yearsDone INTEGER)
pid	INTEGER	"pid" INTEGER
hobbyName	TEXT	"hobbyName" TEXT
yearsDone	INTEGER	"yearsDone" INTEGER
hobby		CREATE TABLE hobby (name TEXT,kind TEXT,setting TEXT)
name	TEXT	"name" TEXT
kind	TEXT	"kind" TEXT
setting	TEXT	"setting" TEXT
people		CREATE TABLE people (id INTEGER,first_name TEXT,last_name TEXT,age INTEGER)
id	INTEGER	"id" INTEGER
first_name	TEXT	"first_name" TEXT
last_name	TEXT	"last_name" TEXT
age	INTEGER	"age" INTEGER
profession		CREATE TABLE profession (name TEXT,field TEXT)
name	TEXT	"name" TEXT
field	TEXT	"field" TEXT
studdied		CREATE TABLE studdied (pid INTEGER,instName TEXT,major TEXT,degreeType TEXT)
pid	INTEGER	"pid" INTEGER
instName	TEXT	"instName" TEXT
major	TEXT	"major" TEXT
degreeType	TEXT	"degreeType" TEXT
worksAs		CREATE TABLE worksAs (pid INTEGER,professionName TEXT,company TEXT,yearsExp INTEGER)
pid	INTEGER	"pid" INTEGER
professionName	TEXT	"professionName" TEXT
company	TEXT	"company" TEXT
yearsExp	INTEGER	"yearsExp" INTEGER

View of Raw DB

ER Diagram


Hobby
name
kind
setting
hasHobby
yearsDone
People
id
first_name
last_name
age
studied
major
degreeType
worksAs
company
yearsExp
Institution
name
type
Profession
name
field
Visual Paradigm Online Free Edition

Example of Randomization

```
# create a list of random people
def randNames(f_names, l_names):
    f_names = f_names.copy()
    l_names = l_names.copy()
    names = []
    id = 0
    while(len(f_names) > 0):
        f_index = random.randint(0, len(f_names) - 1)
        l_index = random.randint(0, len(l_names) - 1)
        f = f_names.pop(f_index)
        l = l_names.pop(l_index)
        age = random.randint(18,50)
        names.append((id,f,l,age))
        id = id + 1
    return names

# people ready
people = randNames(f_names, l_names)
```

Populate the database with random values ->

```python
import os
import sqlite3
import random

def prepDB():
    # delete the db to start over
    os.remove('notebook.db')
    # open db and obtain cursor
    con = sqlite3.connect('notebook.db')
    cur = con.cursor()
    # create tables
    try:
        # people
        cur.execute("""CREATE TABLE people (
        id INTEGER,
        first_name TEXT,
        last_name TEXT,
        age INTEGER
        )
        """)
    except:
        print('table already made')
    try:
        # hobby
        cur.execute("""CREATE TABLE hobby (
        name TEXT,
        kind TEXT,
        setting TEXT
        )
        """)
    except:
        print('table already made')
```

example of people and hobby table ^

```python
def populateDB(people, hobbies, institutions, professions, companies):
    # connect to db
    con = sqlite3.connect('notebook.db')
    cur = con.cursor()
    # insert people
    cur.executemany("insert into people values (?, ?, ?, ?)", people)
    # insert hobbies
    cur.executemany("insert into hobby values (?, ?, ?)", hobbies)
    # insert institutions
    cur.executemany("insert into institution values (?, ?)", institutions)
    # insert professions
    cur.executemany("insert into profession values (?, ?)", professions)
    # iterate through people and create relations
    for person in people:
        # hobby
        hobbyIndex = random.randint(0,len(hobbies)-1)
        hobby = hobbies[hobbyIndex]
        h_years = random.randint(1,10)
        cur.execute("insert into hasHobby values (?,?,?)",
                (person[0], hobby[0], h_years ))
        # profession
        p_i = random.randint(0,len(professions)-1)
        p_years = random.randint(1,10)
        c_name = random.choice(companies)
        cur.execute("insert into worksAs values (?, ?, ?, ?)",
                (person[0], professions[p_i][0], c_name, p_years))
        # institution
        i_i = random.randint(0, len(institutions)-1)
        major = random.choice(['english','computer science',
            'math', 'painting',
            'singing', 'biology',
            'mech E', 'data science',
            'drama', 'creative writing',
            'chemistry', 'physics'])
        degree = random.choice(['bachelors','masters','associates','phd'])
        cur.execute("insert into studdied values (?, ?, ?, ?)",
                (person[0], institutions[i_i][0], major, degree))

    # commit changes
    con.commit()
    con.close()
    print('database populated')

if __name__ == "__main__":
    prepDB()
    populateDB(people,hobbies, institutions, professions, companies)
```


Making the DB

Jupyter Testing

In [3]:
```
import sqlite3
import os
con = sqlite3.connect('notebook.db')
cur = con.cursor()
```

In [34]:
```
h_name = "fishing"

cur.execute("""select first_name, last_name
    from people, hasHobby
    Where id = pid
        And hobbyName = :hName""",
    { 'hName' : h_name}).fetchall()
```

Out[34]:
```
[('john', 'lopez'), ('thomas', 'edwards'), ('stephanie', 'johnson')]
```

In [40]:
```
def hobbyQuery(h_name, h_kind, h_setting):
    query = """
    select first_name, last_name, age
    from people, hasHobby, hobby
    Where id = pid AND hobbyName = name
    """
    if h_name == "":
        pass
    else:
        query += "AND name = '{}'".format(h_name)
    if h_kind == "":
        pass
    else:
        query += "AND kind = '{}'".format(h_kind)
    if h_setting == "":
        pass
    else:
        query += "AND setting = '{}'".format(h_setting)
    return query

query = hobbyQuery('fishing', '', '')
print(query)
cur.execute(query).fetchall()
```

Python Queries

```
# hobby
def hobbyQuery(h_name, h_kind, h_setting):
    query = """
    select first_name, last_name, age
    from people, hasHobby, hobby
    Where id = pid AND hobbyName = name
    """
    if h_name == "":
        pass
    else:
        query += "AND name = '{}'".format(h_name)
    if h_kind == "":
        pass
    else:
        query += "AND kind = '{}'".format(h_kind)
    if h_setting == "":
        pass
    else:
        query += "AND setting = '{}'".format(h_setting)
    return query
```

```python
# all queries
def queryPrefs(minAge,maxAge,pName,pCompany,pField,
        hName,hKind,hSetting,eName,eType,eMajor,eDegree):
    # establish connection with database
    con = sqlite3.connect('notebook.db')
    cur = con.cursor()
    # handle empty int form values
    if minAge == '':
        minAge = -1
    else:
        minAge = int(minAge)
    if maxAge == '':
        maxAge = -1
    else:
        maxAge = int(maxAge)
    # set matches to impossible value
    matches = [-1]
    # check if age was queried
    if minAge > 0 or maxAge > 0:
        # get query
        age_q = ageQuery(minAge, maxAge)
        # get people
        age_approved = cur.execute(age_q).fetchall()
        # intersection with other query was null, break
        if matches == []:
            return []
        # first query triggered
        elif matches == [-1]:
            matches = age_approved
        else:
            # intersection of matches
            matches = set.intersection(set(matches), set(age_approved))
    # check for profession queries
    if pName != "" or pCompany != "" or pField != "":
        job_q = jobQuery(pName, pCompany, pField)
        job_approved = cur.execute(job_q).fetchall()
        if matches == []:
            return []
        elif matches == [-1]:
            matches = job_approved
        else:
            matches = set.intersection(set(matches), set(job_approved))
    # check for hobby queries
    if hName != "" or hKind != "" or hSetting != "":
        hobby_q = hobbyQuery(hName, hKind, hSetting)
```

query.py[+] 83:48

```python
    # check for hobby queries
    if hName != "" or hKind != "" or hSetting != "":
        hobby_q = hobbyQuery(hName, hKind, hSetting)
        hobby_approved = cur.execute(hobby_q).fetchall()
        if matches == []:
            return []
        elif matches == [-1]:
            matches = hobby_approved
        else:
            matches = set.intersection(set(matches), set(hobby_approved))
    # check for education queries
    if eName != "" or eType != "" or eMajor != "" or eDegree != "":
        prof_q = educationQuery(eName, eType, eMajor, eDegree)
        prof_approved = cur.execute(prof_q).fetchall()
        if matches == []:
            return []
        elif matches == [-1]:
            matches = prof_approved
        else:
            matches = set.intersection(set(matches), set(prof_approved))
    # check if there was no user input query
    if matches == [-1]:
        matches = cur.execute("select first_name, last_name, age from people"
).fetchall()
    con.commit()
    con.close()
    return matches
```

NORMAL main query.py[+] utf-8 python 83% 124:23

CASE SCHOOL OF ENGINEERING
CASE WESTERN RESERVE UNIVERSITY

Combining Queries

```python
# module imports
from flask import Flask, render_template, request
import os
import sqlite3
# local imports
from formPref import PreferenceForm
from query import queryPrefs

app = Flask(__name__)
# for flask-wtf need secret key
app.config['SECRET_KEY'] = os.urandom(16)

# generate a list of all people (for testing purposes)
def allPeople():
    con = sqlite3.connect('notebook.db')
    cur = con.cursor()
    people = cur.execute("select * from people").fetchall()
    con.commit()
    con.close()
    return people
# home
@app.route('/')
def home():
    """ Render information on the project """
    return render_template('home.html')

# preferences
@app.route('/preferences', methods=["GET","POST"])
def preferences():
    """ Take in user preferences and return matches """
    pForm = PreferenceForm()
    if request.method == "POST":
        # populate from pForm
        minAge = request.form.get('minAge')
        maxAge = request.form.get('maxAge')
        pName = request.form.get('pName')
        pCompany = request.form.get('pCompany')
        pField = request.form.get('pField')
        hName = request.form.get('hName')
        hKind = request.form.get('hKind')
        hSetting = request.form.get('hSetting')
```

main.py 7:7

```python
@app.route('/preferences', methods=["GET","POST"])
def preferences():
    """ Take in user preferences and return matches """
    pForm = PreferenceForm()
    if request.method == "POST":
        # populate from pForm
        minAge = request.form.get('minAge')
        maxAge = request.form.get('maxAge')
        pName = request.form.get('pName')
        pCompany = request.form.get('pCompany')
        pField = request.form.get('pField')
        hName = request.form.get('hName')
        hKind = request.form.get('hKind')
        hSetting = request.form.get('hSetting')
        eName = request.form.get('eName')
        eType = request.form.get('eType')
        eMajor = request.form.get('eMajor')
        eDegree = request.form.get('eDegree')
        people = queryPrefs(minAge,
                maxAge,
                pName,
                pCompany,
                pField,
                hName,
                hKind,
                hSetting,
                eName,
                eType,
                eMajor,
                eDegree)
        return render_template(
                'matches.html',
                people = people
                )
    return render_template(
            'preferences.html',
            form=pForm,
            )

if __name__ == "__main__":
    app.run(debug=True)
```

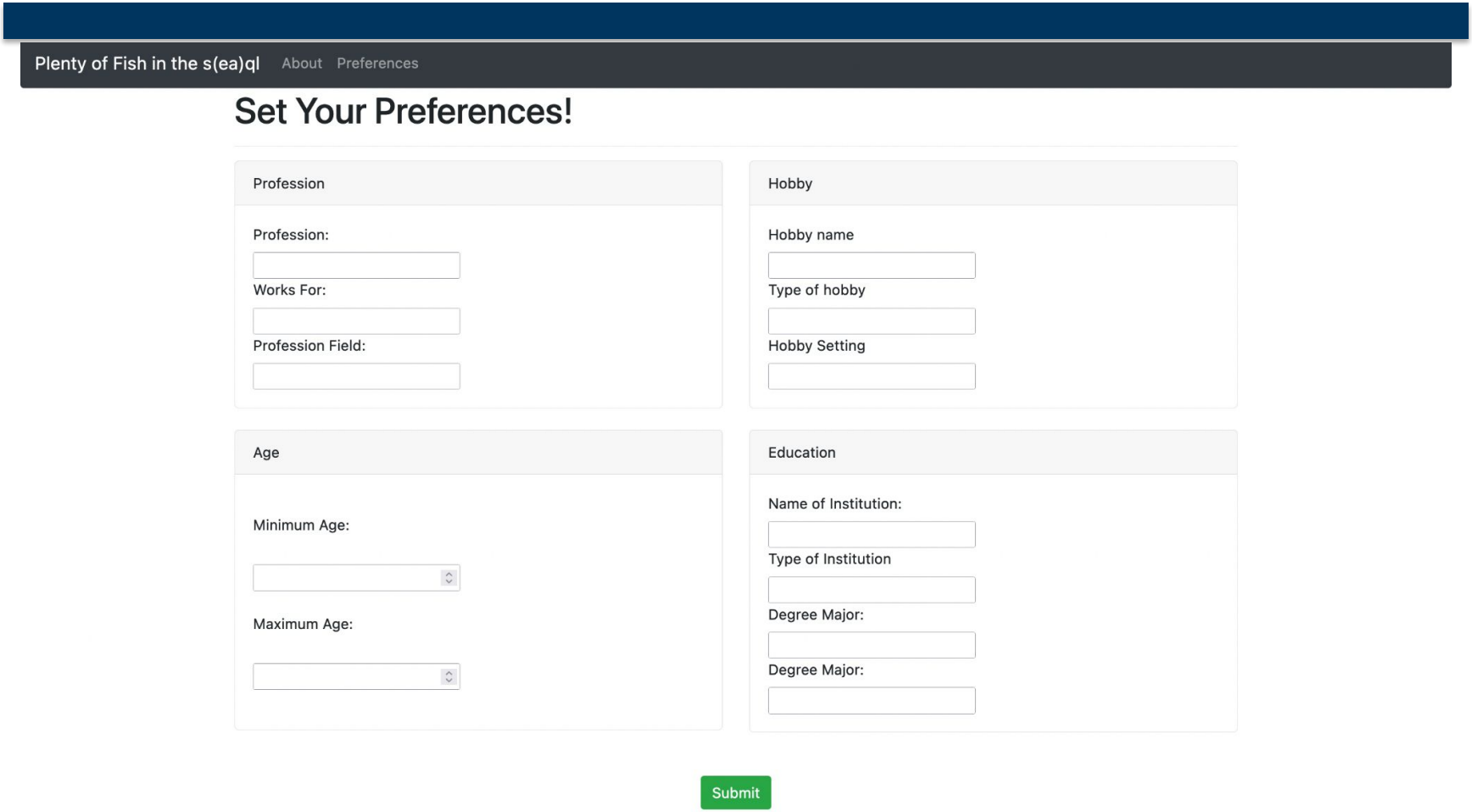
NORMAL main main.py utf-8 python 54% 37:23

Flask

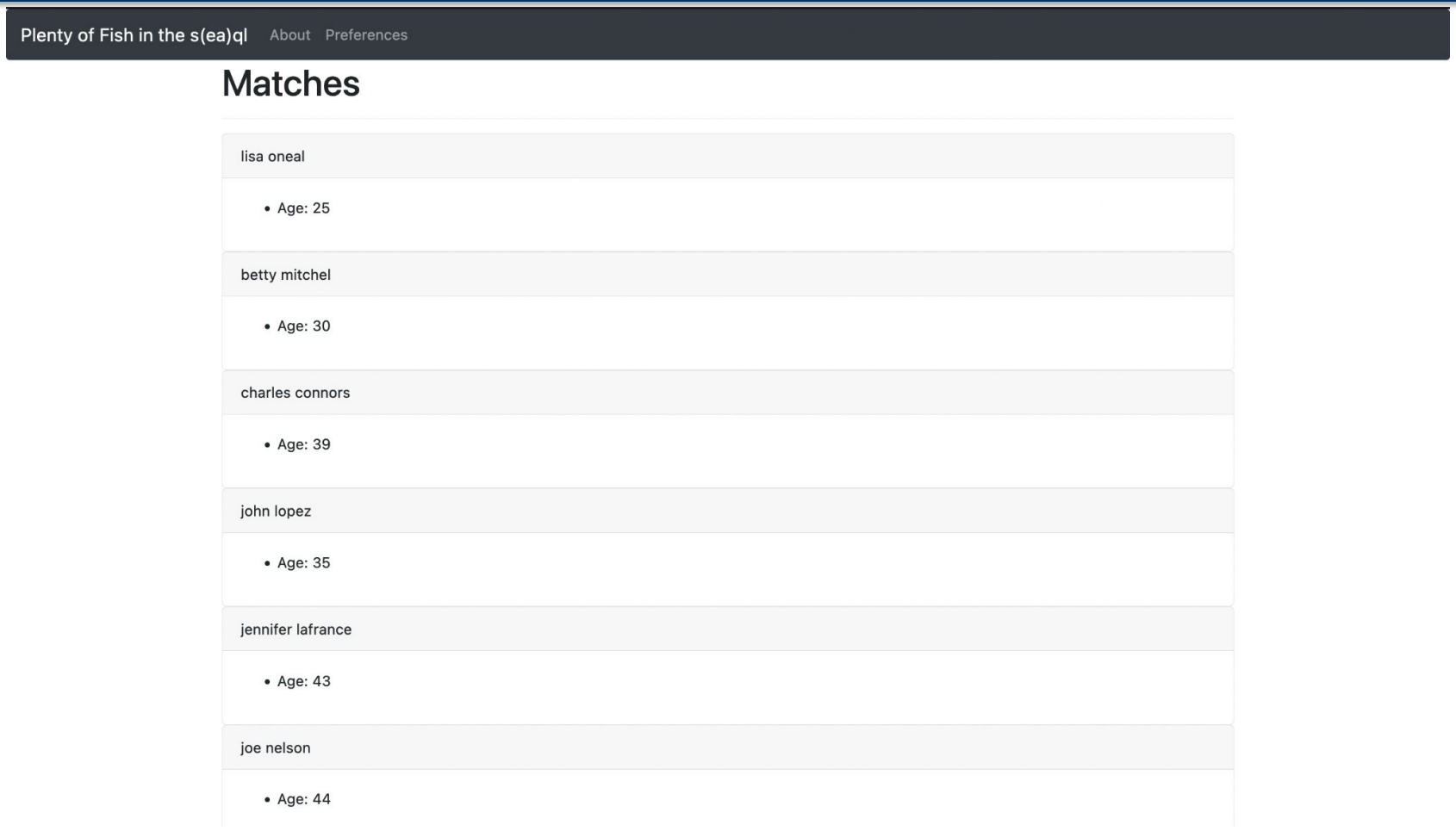
Jinja Templating

```
{% extends "base.html" %}

{% block title %} Matches {% endblock %}

{% block content %}

  <h1> Matches </h1>
  <hr>

  <h1> {{ minAge }} </h1>
  <h1> {{ maxAge }} </h1>
  <h1> {{ state }} </h1>
  <h1> {{ city }} </h1>
  <h1> {{ pName }} </h1>
  <h1> {{ pField }} </h1>
  <h1> {{ eLevel }} </h1>
  <h1> {{ eMajor }} </h1>

  <div class="row">
  {% for person in people %}
    <div class="col-md-12">
      <div class="card">
        <div class="card-header"> {{person[0] + " " + person[1]}} </div>
        <div class = "card-body">
          <ul>
            <li> Age: {{person[2]}} </li>
          <ul>
        </div>
      </div>
    </div>
  {% endfor %}
  </div>

{% endblock %}
```

UI Form


Plenty of Fish in the s(ea)ql
About
Preferences
Set Your Preferences!
Profession
Profession:
Works For:
Profession Field:
Hobby
Hobby name
Type of hobby
Hobby Setting
Age
Minimum Age:
Maximum Age:
Education
Name of Institution:
Type of Institution
Degree Major:
Degree Major:
Submit

Matches UI


Plenty of Fish in the s(ea)ql
About
Preferences
Matches
lisa oneal
• Age: 25
betty mitchel
• Age: 30
charles connors
• Age: 39
john lopez
• Age: 35
jennifer lafrance
• Age: 43
joe nelson
• Age: 44