Homework 1

by Movsisyan Mher

1. Relational schema design (5 pts)

In the first session, we ran a survey and collected your responses. The file “Database-Welcome.xlsx” is uploaded to Moodle, in the current assignment.

Use the data from the “Database-Welcome.xlsx” file to define relational schemas.
Define the schemas in the relational style, similar to the example used in class S = (ID, name, department).
For every attribute in the schema, specify its domain.
For each schema specify the primary key that you chose. Explain why it works as a primary key.

In [16]:
```
import pandas as pd
import numpy as np

# readin data
df = pd.read_excel('data\Database-Welcome.xlsx')

df.head()
```

Out[16]:

	Date	Session	Voter	Welcome to DS 205!:	What are your interests out of studies?:	Which year of studies are you in?:	Which one describes your knowledge and experience with databases?:	Which programming languages do you know?:	What are your expectations from this course?: 1	Can you bring a laptop computer to the class?:	Thank you:
0	2022-01-20	4	4	Heart	Basketball Drawing_portraits Coding	3rd (Junior)	Intermediate	Python Julia	Pass this course\nIf possible learn some funda...	Yes	Heart
1	2022-01-20	4	5	NaN	NaN	2nd (Sophomore)	Advanced	Python T-Sql	Gain advanced knowledge of databases from A-Z	Yes	NaN
2	2022-01-20	4	6	Heart	Reading Piano Classical_music	2nd (Sophomore)	Beginner	Python R JS	NaN	Yes	Heart
3	2022-01-20	4	7	NaN	Playing_the_guitar Hiking	2nd (Sophomore)	Beginner	Python JS	NaN	Yes	NaN
4	2022-01-20	4	8	NaN	NaN	NaN	Beginner	Python CSharp	NaN	Yes	Heart

Dropping unnecessary columns

In [17]:
```
df.drop(["Date", "Session", "Welcome to DS 205!:", "Thank you:"], axis=1, inplace=True)
df.head()
```

Out[17]:

	Voter	What are your interests out of studies?:	Which year of studies are you in?:	Which one describes your knowledge and experience with databases?:	Which programming languages do you know?:	What are your expectations from this course?: 1	Can you bring a laptop computer to the class?:
0	4	Basketball Drawing_portraits Coding	3rd (Junior)	Intermediate	Python Julia	Pass this course\nIf possible learn some funda...	Yes
1	5	NaN	2nd (Sophomore)	Advanced	Python T-Sql	Gain advanced knowledge of databases from A-Z	Yes
2	6	Reading Piano Classical_music	2nd (Sophomore)	Beginner	Python R JS	NaN	Yes
3	7	Playing_the_guitar Hiking	2nd (Sophomore)	Beginner	Python JS	NaN	Yes
4	8	NaN	NaN	Beginner	Python CSharp	NaN	Yes

Defining a schema for the survey results:

$$Voters = (ID, yearOfStudy, priori, expectation, laptop)$$

Voters.year_of_study is not timeless, but I'll ignore that. It is the "Which year of studies are you in?:" column.
Voters.ID is a primary key, it is the "Voter" column in the data because it is unique and an obvious choice.
Voters.priori is the level of knowledge and experience of the voter, is the "Which one describes your knowledge and experience with databases?:" column in the data.
Voters.expectation is the level of expectation of the voter, is the "How much do you expect to learn from this course?:" column in the data.
Voters.laptop is the "Can you bring a laptop computer to the class?:" column in the data.

$$Interests = (ID, voter, interest)$$

Interests.ID is a primary key, it is generated from the segmentation process.
Interests.voter is a foreign key that references Voters.ID

$$LanguageKnowers = (ID, voter, language)$$

LanguageKnowers.ID is a primary key, it is generated from the segmentation process.
LanguageKnowers.voter is a foreign key that references Voters.ID
LanguageKnowers.language is the individual cleaned-up language from the "Which programming languages do you know?:" column in the data

2. Data clean up (4 pts)

Think of a method to identify the data items that have the same meaning. For example, JS and JavaScript and Java script have the same semantics, however written in slightly different ways. Describe your approach of solving this issue in simple words.
Design a function that receives a token (e.g. JS) as the input and returns an output which is unique for all the variants.

For example, given the function f(x) it should require that:

$$f(''JavaScript'') = f(''JavaScript'') = f(''JS'') = ''JS''$$

and

$$f(''py'') = f(''Python'') = f(''python'') = ''Python''$$

and so on.
Implement the body of the function in your preferred language, or using pseudo-code.

In [19]:
```python
# getting (almost) unique entries for the known languages
p_lang_entries = [[*i.split(' ')] for i in df.iloc[:, 4].fillna('None')]
p_langs = pd.Series([x for l in p_lang_entries for x in l]).str.lower()
p_langs.value_counts()
```

Out[19]:
```
python        16
r              8
javascript     4
js             3
none           3
julia          1
t-sql          1
csharp         1
c_sharp        1
java           1
dtype: int64
```

In [20]:
```python
from fuzzywuzzy import fuzz
import warnings

warnings.filterwarnings("ignore")

# creating a fuzz-ratio matrix of already known languages
lang_fuzz = pd.DataFrame(columns = p_langs.unique(), index = p_langs.unique()).fillna(0)

# filling the matrix with the fuzzy-ratio values
for i, col in lang_fuzz.iteritems():
    for j, row in col.iteritems():
        lang_fuzz.loc[i, j] = fuzz.ratio(i, j)

lang_fuzz
```

Out[20]:

	python	julia	t-sql	r	js	csharp	c_sharp	javascript	none	java

	python	julia	t-sql	r	js	csharp	c_sharp	javascript	none	java
python	100	0	18	0	0	17	15	25	40	0
julia	0	100	20	0	29	18	17	27	0	44
t-sql	18	20	100	0	29	18	17	13	0	0
r	0	0	0	100	0	29	25	18	0	0
js	0	29	29	0	100	25	22	33	0	33
csharp	17	18	18	29	25	100	92	38	0	20
c_sharp	15	17	17	25	22	92	100	35	0	18
javascript	25	27	13	18	33	38	35	100	0	57
none	40	0	0	0	0	0	0	0	100	0

Writing the function

In [21]:
```
languages = [
    "python", "csharp", "julia",
    "none", "java", "javascript",
    "r", "scala", "matlab", "go",
    "c", "c++", "ruby", "perl", "py",
    "php", "swift", "haskell", "js","lua",
    "rust", "visual_basic", "sql", # gonna treat t-sql, postgresql, mysql as the same
    "assembly", "ecmascript", "c#"
    ]

def match_lang(x: str, threshold: int = 74, known: list = languages):
    """Matches a string to a given programming language. If it fails
    to do so, it creates a new entry in the list of languages"""

    #
    if x == "" or str(x) == "None":
        return "none"

    # making it lowercase
    lower_x = x.lower()
    # creating a dict that will store the fuzzy ratios
    scores = {i:0 for i in known}

    # checking fuzzy ratio for each registered language
    for lang in known:
        scores[lang] = fuzz.ratio(lang, lower_x)

    # print("Best match:", max(scores, key=scores.get), max(scores.values()))

    if max(scores.values()) < threshold:
        # print("Adding new language:", x)
        known.append(x)
        # insert new language into the languages table

        return x
    else:
        # get best match
        best_x = max(scores, key=scores.get)

        # check if it is a known short form
        if best_x == "js" or best_x == "ecmascript":
            return "javascript"
        elif best_x == "py":
            return "python"
        elif best_x == "csharp":
            return "c#"

        # return the best match
        return best_x
```

In [22]:
```
match_lang("r")
```

Out[22]:
```
'r'
```

In [23]:
```
match_lang("js")
```

Out[23]:
```
'javascript'
```

In [24]:
```
match_lang("jaAVASCRIPT")
```

```
Out[24]: 'javascript'
```

```
In [25]: match_lang("C#")
```

```
Out[25]: 'c#'
```

```
In [26]: match_lang("py")
```

```
Out[26]: 'python'
```

3. Data population (2 pts)

Visualize the schema instances as tables and populate the data from the Excel file into the tables as rows, so that they obey the relational schema format

Creating and visualizing the Languages table

```
In [27]:
LanguageKnowers = pd.DataFrame({"ID": [], "voter": [], "language": []})
Interests = pd.DataFrame({"ID": [], "voter": [], "interest": []})

df.fillna('None', inplace=True)
lang_id = 0
interest_id = 0

for voter in df.iterrows():
    for lang in voter[1][4].split(" "):
        match = match_lang(lang)
        if match != "none":
            LanguageKnowers = LanguageKnowers.append({"ID": lang_id, "voter": voter[1][0], "language": mat
            lang_id += 1

    for interest in voter[1][1].split(" "):
        match = match_lang(interest, known=["none"])
        if match != "none":
            Interests = Interests.append({"ID": interest_id, "voter": voter[1][0], "interest": match}, ign
            interest_id += 1

LanguageKnowers.voter = LanguageKnowers.voter.astype(int)
LanguageKnowers.ID = LanguageKnowers.ID.astype(int)
LanguageKnowers.set_index("ID", inplace=True)

Interests.voter = Interests.voter.astype(int)
Interests.ID = Interests.ID.astype(int)
Interests.set_index("ID", inplace=True)
```

```
In [28]: Interests
```

Out[28]:

	voter	interest
ID		
0	4	Basketball
1	4	Drawing_portraits
2	4	Coding
3	6	Reading
4	6	Piano
5	6	Classical_music
6	7	Playing_the_guitar
7	7	Hiking
8	9	Working_out
9	9	Doing_projects
10	10	Reading_books
11	10	Growing_plants
12	10	Writing_short_stories
13	11	Reading
14	11	Playing_the_piano_singing
15	11	Volleyball

	voter	interest
ID		
16	12	talk_with_friends
17	12	eat
18	12	walk
19	14	Reading
20	14	Movies
21	14	Learning_new_things
22	16	Chess
23	16	Reading
24	16	Languages
25	17	Data_analysis
26	17	Reading
27	17	Working_out
28	18	Solving_puzzles
29	18	Watching_Criminal_Cases
30	18	Watching_Medical_Cases
31	19	Data_analytics
32	19	QA
33	19	Music
34	20	Reading

In [29]:
```
LanguageKnowers
```

Out[29]:

	voter	language
ID		
0	4	python
1	4	julia
2	5	python
3	5	sql
4	6	python
5	6	r
6	6	javascript
7	7	python
8	7	javascript
9	8	python
10	8	c#
11	9	python
12	9	c#
13	9	javascript
14	10	python
15	10	r
16	11	javascript
17	11	python
18	11	r
19	12	python
20	12	r
21	12	javascript
22	13	python
23	13	r

	voter	language
ID		
24	14	python
25	16	python
26	16	r
27	17	r
28	17	python
29	17	java
30	18	python
31	18	r
32	18	javascript
33	19	python

In [30]:

```
Voters = df.iloc[:, [0, 2, 3, 5, 6]].set_axis(["ID", "yearOfStudy", "priori", "experience", "laptop"], axi
Voters
```

Out[30]:

	yearOfStudy	priori	experience	laptop
ID				
4	3rd (Junior)	Intermediate	Pass this course\nIf possible learn some funda...	Yes
5	2nd (Sophomore)	Advanced	Gain advanced knowledge of databases from A-Z	Yes
6	2nd (Sophomore)	Beginner	None	Yes
7	2nd (Sophomore)	Beginner	None	Yes
8	None	Beginner	None	Yes
9	2nd (Sophomore)	Intermediate	Getting more experience dealing with non-relat...	Yes
10	2nd (Sophomore)	Beginner	I strongly believe that the course will help m...	Yes
11	2nd (Sophomore)	Intermediate	None	Yes
12	3rd (Junior)	Intermediate	learn sql and hopefully pass)	Yes
13	3rd (Junior)	Intermediate	None	Yes
14	2nd (Sophomore)	Beginner	Advance my knowledge in programming and data s...	Yes
15	None	None	None	None
16	3rd (Junior)	Beginner	Learn new skills	Yes
17	3rd (Junior)	Intermediate	None	None
18	3rd (Junior)	Beginner	Learn SQL	Yes
19	4th (Senior)	Beginner	A	Yes
20	2nd (Sophomore)	Beginner	Understandable material	No
21	None	Beginner	yes	Yes
22	None	None	None	Yes

4. Relational Algebra queries (5 pts)

Given the relational schemas you defined in Problem 1, write queries in relational algebra language to:

- a) Find voters that are sophomores or juniors
- b) Find the unique list of interests that voters named
- c) Find the voters that named JavaScript as a programming language they know

Answers:

- a) select * from Voters where yearOfStudy = "2nd (Sophomore)" or 3rd (Junior)
- b) select distinct interest from Interests
- c) select * from Voters where ID in (select voter from LanguageKnowers where language = "javascript")

5. Foreign keys (4 pts)

Specify the foreign keys between the tables, as well as the referencing and referenced tables.

Answers:

- Interests.voter is a foreign key that references Voters.ID
- LanguageKnowers.voter is a foreign key that references Voters.ID
- Voters.ID is a primary key that is referenced by the above mentioned tables (Interests, LanguageKnowers)