Assignment No. 2 – SQL & Data Frames and Deductive Databases

Objectives of this assignment

This assignment will reinforce fundamental ideas related to:

- Relational Algebra
- Pandas Data Frames & SQL
- Deductive Databases

For the Deductive Databases portion of the assignment we will use a Logic Programming language (**Prolog**) instead of a true Deductive DB system. Having said that, the syntax of the language component in both, Datalog and Prolog, is very similar, hence, it is acceptable from the academic perspective.

Homework Description

You will be given eight (8) **relations**, stored in eight individual `.csv` files (one relation per file). For example, here is a view of one of those files ("workson.csv"):

NAME	PROJECT	EFFORT
john	productx	32
john	producty	8
ramesh	productz	40
joyce	productx	20
joyce	producty	20
franklin	producty	10
franklin	productz	10
franklin	computeriza	10
franklin	reorganizatic	10
alicia	newbenefits	30
alicia	computeriza	10
ahmad	computeriza	35
ahmad	newbenefits	5
jennifer	newbenefits	20
jennifer	reorganizatic	15
james	reorganizatic	10

This relation describes the effort associated to a given project, by a given employee. For example, "john" spend 32 hours a week working on *project* "productx" and 8 hours a week on *project* "producty".

Part 1: Data Frames and SQL Queries (5 points)

- Using/importing `pandas` and `sqlite3` modules/packages, you will write a number of SQL queries against the given database. The former package will give you access to data frames whilst the latter one will enable your code to run sql-queries from Python. See the examples provided in class as a guideline.
- The current definitions in the `.csv` files provided, are not favourable for proper **normalization**. However, for the purpose of this assignment you will **not** be asked to

normalize the entities/relationships. You can use the relations "as is".

Here are the queries you must write:

```
QUERIES

Q1: Are there any employees that are "female" and that work in project
"computerization" with an effort of "10" hours per week, and that have
"jennifer" as a supervisor? If so, list them.

Q2: Who is the employee who makes over "40000" dollars a year and
works for the "research" department?

Q3: Who is the supreme chief of this fictional company (aka the Preside

Q4: Who are the individuals that work on project "productx" with an *ef
of 20 or more hours a week?
```

Part 2: Deductive Databases Queries (5 points)

You have been given one `Prolog script` representing a deductive database `companyext.pl`. Please make note that **you will have to add some rules to those already in the Prolog script to make everything work** and be able to write the queries mentioned below.

```
    Script: companyext.pl

    Queries: you are allowed to use all facts and rules in the system, and

    QUERIES (same as above)

    Q1: Are there any employees that are "female" and that work in project
    "computerization" with an effort of "10" hours per week, and that have
    "jennifer" as a supervisor? If so, list them.

    Q2: Who is the employee who makes over "40000" dollars a year and works
    for the "research" department?

    Q3: Who is the supreme chief of this fictional company (aka the Preside

    Q4: Who are the individuals that work on project "productx" with an *ef
    of 20 or more hours a week?
```

Here is a snapshot of a portion of the `companyext` database:

```
project(productx).
project(producty).
project(productz).
project(computerization).
project(reorganization).
project(newbenefits).

works_on(john, productx, 32).
works_on(john, producty, 8).
works_on(ramesh, productz, 40).
works_on(joyce, productx, 20).
works_on(joyce, producty, 20).
works_on(franklin, producty, 10).
works_on(franklin, productz, 10).
works_on(franklin, computerization, 10).
works_on(franklin, reorganization, 10).
works_on(alicia, newbenefits, 30).
works_on(alicia, computerization, 10).
works_on(ahmad, computerization, 35).
works_on(ahmad, newbenefits, 5).
works_on(jennifer, newbenefits, 20).
works_on(jennifer, reorganization, 15).
works_on(james, reorganization, 10).
```

Part 3: Reflection (5 points)

Based on your experience writing the requested queries in Part 1 & Part 2:

```
1. What is the most salient point between the commonalities and differences
of queries formulated in SQL and those expressed in Prolog?
2. Did you follow the same 'logic' when writing the queries in both languag
3. What are the differences between the way the "data" and the "queries" ar
represented in SQL?
4. What are the differences between the way the "data" and the
"rules/queries" are represented in Prolog?
```

Appendix I - Pandas' Data Frames & SQL (a sample of material distributed through D2L)

The file containing the portion of code shown below, was distributed as a .zip file under `Lesson 5` : **pandas-sql**. The `jupyter notebook` inside was named **relational_data.ipynb**.

```
### Create a DB with "Person" and "Grades" tables.
import sqlite3
conn = sqlite3.connect("database.db")
cursor = conn.cursor()

### when you are done, call conn.close()

cursor.execute("""
CREATE TABLE person (
    id INTEGER PRIMARY KEY,
    last_name TEXT,
    first_name TEXT
);""")

cursor.execute("""
CREATE TABLE grades (
    person_id INTEGER PRIMARY KEY,
    hw1_grade INTEGER,
    hw2_grade INTEGER
);""")

conn.commit()

# Inserting data
cursor.execute("INSERT INTO person VALUES (1, 'Kolter', 'Zico');")
cursor.execute("INSERT INTO person VALUES (2, 'Xi', 'Edgar');")
```

```
cursor.execute("INSERT INTO person VALUES (3, 'Lee', 'Mark');")
cursor.execute("INSERT INTO person VALUES (4, 'Mani', 'Shouvik');")
cursor.execute("INSERT INTO person VALUES (5, 'Gates', 'Bill');")
cursor.execute("INSERT INTO person VALUES (6, 'Musk', 'Elon');")

cursor.execute("INSERT INTO grades VALUES (5, 85, 95);")
cursor.execute("INSERT INTO grades VALUES (6, 80, 60);")
cursor.execute("INSERT INTO grades VALUES (100, 100, 100);")

# It can be handy to dump the results of a query directly into a Pandas
# DataFrame. Fortunately, Pandas provides a nice call for doing this, the
# pd.read_sql_query() function, with takes the database connection and an
# optional argument to set the index of the Pandas dataframe to be one of t
# columns. See below:

pd.read_sql_query("SELECT * from person;", conn, index_col="id")

# And here there is an example of an inner join:

pd.read_sql_query("SELECT * FROM person, grades WHERE person.id = grades.pe
```

Please consult the material provided in D2L and in class for full details.

Appendix II - Company Database in Prolog

Here is the company database in Prolog format. We have included all `facts`, and have also provided two simple and `non-recursive rules`.

```
%-----------------------------------------------------
% Facts

employee(john).
employee(franklin).
employee(alicia).
employee(jennifer).
employee(ramesh).
employee(joyce).
employee(ahmad).
employee(james).

salary(john, 30000).
salary(franklin, 40001).
salary(alicia, 25000).
```

```
salary(jennifer, 43000).
salary(ramesh, 38000).
salary(joyce, 25000).
salary(ahmad, 25000).
salary(james, 55000).

department(john, research).
department(franklin, research).
department(alicia, administration).
department(jennifer, administration).
department(ramesh, research).
department(joyce, research).
department(ahmad, administration).
department(james, headquarters).

supervise(franklin, john).
supervise(franklin, ramesh).
supervise(franklin, joyce).
supervise(jennifer, alicia).
supervise(jennifer, ahmad).
supervise(james, franklin).
supervise(james, jennifer).

female(alicia).
female(jennifer).
female(joyce).

male(john).
male(franklin).
male(ramesh).
male(ahmad).
male(james).

project(productx).
project(producty).
project(productz).
project(computerization).
project(reorganization).
project(newbenefits).

works_on(john, productx, 32).
works_on(john, producty, 8).
works_on(ramesh, productz, 40).
works_on(joyce, productx, 20).
works_on(joyce, producty, 20).
```

```
works_on(franklin, producty, 10).
works_on(franklin, productz, 10).
works_on(franklin, computerization, 10).
works_on(franklin, reorganization, 10).
works_on(alicia, newbenefits, 30).
works_on(alicia, computerization, 10).
works_on(ahmad, computerization, 35).
works_on(ahmad, newbenefits, 5).
works_on(jennifer, newbenefits, 20).
works_on(jennifer, reorganization, 15).
works_on(james, reorganization, 10).

%-----------------------------------------------------

% Rules
% X is superior to Y in the company hierarchy.
superior(X,Y) :- supervise(X,Y).

% X is subordinate to Y in the company hierarchy.
subordinate(X,Y) :- superior(Y,X).

% Note that these two queries, as written, are NOT recursive.
```

Appendix III - Prolog Code Example: Greek Gods

Here is a sample Prolog code representing the Greek Gods Family Tree. This code may help you to figure out how to write your own code, especially if recursion were involved.

```
% Greek Gods Family Tree

% parent(X, Y)  This states that x is a parent of y.
% male(X)  This states if x is a male.
% female(X) This states if x is a female.

% Cronus and Rhea are the parents of Hestia, Pluto, Poseidon, Zeus, Hera,
% and Demeter.
% Zeus is the father of Athena.
% Zeus and Hera are the parents of Ares, Hebe, and Hephaestus.
% Zeus and Demeter are the parents of Persephone.

% isFather(X,Y).
% isMother(X,Y).
% isDaughter(X,Y).
```

```
% isSon(X,Y).

% isAncestor(X, Y).
% ------------------------------------------------------------------------

% % Database facts: % %

parent(cronus,hestia).
parent(cronus,pluto).
parent(cronus,poseidon).
parent(cronus,zeus).
parent(cronus,hera).
parent(cronus,demeter).
parent(rhea, hestia).
parent(rhea, pluto).
parent(rhea, poseidon).
parent(rhea, zeus).
parent(rhea, hera).
parent(rhea, demeter).

parent(zeus, athena).

parent(zeus, ares).
parent(zeus, hebe).
parent(zeus, hephaestus).
parent(hera, ares).
parent(hera, hebe).
parent(hera, hephaestus).

parent(zeus, persephone).
parent(demeter, persephone).


male(cronus).
male(pluto).
male(poseidon).
male(zeus).
male(ares).
male(hephaestus).
female(rhea).
female(hestia).
female(hera).
female(demeter).
female(athena).
female(hebe).
```

```
female(persephone).


%%  Database Rules %%

isFather(X, Y):-
  male(X),
  parent(X, Y).

isMother(X, Y):-
  female(X),
  parent(X, Y).

isDaughter(X, Y):-
  female(X),
  parent(Y, X).

isSon(X, Y):-
  male(X),
  parent(Y, X).

isAncestor(X, Y):- parent(X, Y).
isAncestor(X, Y):- parent(X, T), parent(T, Y).

% Queries Samples

% 1. Using existing predicates write the rule <fullSibling(X,Y)> that retur
% all siblings of a given person.
% Try it finding all siblings of <zeus>.

fullSibling(X,Y):-isFather(F1,X), isMother(M1,X), isFather(F2,Y),
                  isMother(M2,Y), F1 = F2, M1 = M2.


% 2. Using the previously written rule, <fullSibling(X,Y)>, write another r
% Try it with <zeus>.

fullSiblingButOneself(X,Y):-fullSibling(X,Y), not(Y = X).
```

Now, let's look at the results of running the above mentioned queries:

```
Welcome to SWI-Prolog (threaded, 64 bits, version 8.4.3)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- chdir('C:/Users/oappel/Desktop/A2-W2024/A2-W2024/').

?- consult(greekgodsanswers).

?- fullSibling(zeus,Y).
Y = hestia ;
Y = pluto ;
Y = poseidon ;
Y = zeus ;
Y = hera ;
Y = demeter ;
false.

?- fullSiblingButOneself(zeus,Y).
Y = hestia ;
Y = pluto ;
Y = poseidon ;
Y = hera ;
Y = demeter ;
false.

?-
```

What do you need to submit to D2L? (please, only one submission per team)

A zip file containing:

1. Your Jupyter Notebook file for the SQL portion of the assignment.
2. Your Prolog script.
3. A docx or pdf file answering the questions of `Part 3: Reflection` **and** containing screenshots of your program executions, showing that they are delivering on what you were asked to do.

When is it due?

Sunday **17th of March** of 2024 @ midnight.

How many people can work on it?

A maximum of 4 people per team, unless something different has been agreed with your instructor. Individual submissions are welcome, too.