Description
This assignment covers schema refinement and normalization.

Details and Grading
This assignment is made up of 2 parts, collectively worth 80 points, or 5% of the over-all course grade. If this assignment is submitted late, you will receive no credit. This assignment is to be completed individually. Please consult the course syllabus for a description of our academic honesty policy.

Part 1 of the assignment focuses on datalog, it should be implemented in LogiQL, a dialect of datalog that we covered in class. Develop and test your code with https://developer.logicblox.com/playground, referring to the sample code (on the course website), to the LogiQL in 30 minutes tutorial https://developer.logicblox.com/content/docs4/tutorial/repl/section/split.html, and to the language reference manual at https://developer.logicblox.com/documentation/, as appropriate. Name your submission for this part hw5_part1.txt and submit it as a single file.

Part 2 of the assignment focuses on recursive SQL. We should be able to execute the queries you submit in our PostreSQL database. Name your submission for this part hw5_part2.sql and submit it as a single file.

Submission instructions
Submit your assignment to gitlab on gitlab.cci.drexel.edu. I assume that you already followed the steps to create your git repository. I will refer to the root directory of your git repository as $GIT_HOME. Create a directory called cs500-hw5 (case-sensitive, use exactly this name) under $GIT_HOME:

```bash
mkdir $GIT_HOME/cs500-hw5
cd $GIT_HOME/cs500-hw5
```

Place the files you wish to submit, hw5_part1.txt and hw5_part2.sql, into this directory. You can now commit your assignment as follows:

```bash
git add *
git commit -m ‘homework 5’
git push
```

You may submit multiple times before the deadline, only your last submission committed before the deadline will be graded.
Part 1 (60 points): Datalog

Assume that you are given the following base (EDB) relations.

- person (name, gender) - both attributes are of type string; use “M” and “F” to denote gender, name is the candidate key.
- parent_of (parent, child) - parent and child are of type string, and must appear in relation person, (parent, child) is the composite candidate key.

Write LogiQL programs that compute the following:

a. **Siblings:** siblings(x, y), listing all pairs of siblings; if Ann and Bob are siblings, then siblings should include both (“Ann”, “Bob”) and (“Bob”, “Ann”).

b. **Related persons:** related(x,y), listing all pairs of related persons. Ann and Bob are related if Ann is a parent or ancestor of Bob, if they are siblings, or if they are cousins, no matter how far removed. Any relationship between Ann and Bob that can be computed from parent_of counts. As for siblings, include both (“Ann”, “Bob”) and (“Bob”, “Ann”) in the result.

c. **Same generation:** same_generation(x,y,g), listing all pairs of persons who have a common ancestor and are of the same generation g with respect to that ancestor. For a pair of siblings, g = 1.

d. **Descendants by gender:** num_descendants_by_gender(x,g,n), listing, for each person x who has descendants, and for each gender of the descendants g, the number of such descendants n. For example, if Ann has 1 female and 1 male child, 2 female and 3 male grandchildren, and no other descendants, the result should contain 2 tuples for Ann: (“Ann”, “F”, 3) and (“Ann”, “M”, 4).

Your answer for all (a) – (d) should be in the same file called hw5_part1.txt. Include addblock statements as appropriate. Also include test data for persons and parent_of (using exec +person(“Ann”. “F”) etc. See https://www.cs.drexel.edu/~julia/cs500/documents/code/lb_food.txt and https://www.cs.drexel.edu/~julia/cs500/documents/code/lb_person.txt for what your submission should look like.
Part 2 (20 points): Recursive SQL

You may assume that the input is in the same format as in Part 1 above: relations Person (name, gender) and Parent_of (parent, child). Specifically, you should use create table statements below:

```sql
create table Person (  
    name varchar(64) primary key,  
    gender char(1)  
);

create table Parent_of (  
    parent varchar(64),  
    child varchar(64),  
    primary key (parent, child),  
    foreign key (parent) references Person(name),  
    foreign key (child) references Person(name)  
);
```

a. **Same gender descendants**: Compute all pairs of persons such that the first is an ancestor of the second, and they have the same gender. The result should have the schema (ancestor, descendant).

b. **Descendants by gender**: For all persons who have a descendant, compute the number of descendants by gender (the same question as 1(d)). The result should have the schema (ancestor, gender, num_descendants).