COMP 4522-002: Assignment 2 Reflection

Gareth Carvalho

1. What is the most salient point between the commonalities and differences of queries formulated in SQL and those expressed in Prolog?

   Definitely, the biggest difference between the two formats is the syntax for everything. Storing data, retrieving data, joins and other operations. Everything is completely different.

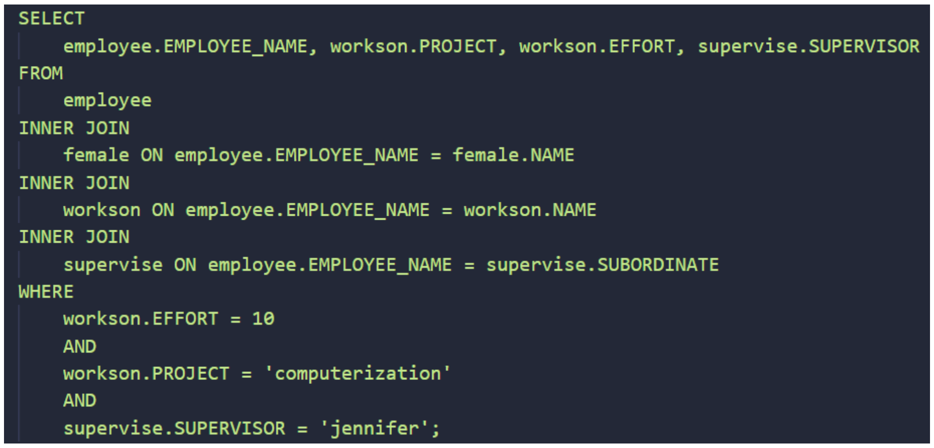
   Also, the way relations are done isn't as explicit in Prolog as in SQL. What I mean is that in SQL we have primary, secondary, and composite keys (not that the tables in this assignment used them). In comparison, in Prolog relations are more implicit. You repeat information in different "tables" like employee names, and the names that repeat across those tables are implicitly linked when querying.

   I could very well be off the mark in these observations, but that's just what it felt like to me. Also, in some ways, Prolog felt a lot simpler than SQL. The rules just felt a lot more intuitive, whereas in SQL, I feel I had to think a lot more about what I wanted the end table to look like, and piece together the right operations to get there.

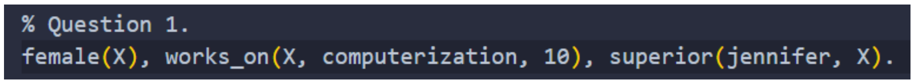
2. Did you follow the same 'logic' when writing the queries in both languages?

   For the most part, I followed similar logic when thinking about the queries. I only realized it afterward, but when I was constructing the SQL query, I constructed it in phases similar to the different parts of my Prolog queries.

   For example, this is my query in SQL and Prolog respectively.



```sql
SELECT
    employee.EMPLOYEE_NAME, workson.PROJECT, workson.EFFORT, supervise.SUPERVISOR
FROM
    employee
INNER JOIN
    female ON employee.EMPLOYEE_NAME = female.NAME
INNER JOIN
    workson ON employee.EMPLOYEE_NAME = workson.NAME
INNER JOIN
    supervise ON employee.EMPLOYEE_NAME = supervise.SUBORDINATE
WHERE
    workson.EFFORT = 10
    AND
    workson.PROJECT = 'computerization'
    AND
    supervise.SUPERVISOR = 'jennifer';
```



```prolog
% Question 1.
female(X), works_on(X, computerization, 10), superior(jennifer, X).
```

The `INNER JOINS` in my SQL query follow the same order as the joins in my Prolog query. This is because that's just how I thought about it while creating the SQL query. Afterwards, when I did the Prolog query, they came together very fast because that's how my brain naturally thought about it.

3. What are the differences between the way the "data" and the "queries" are represented in SQL?

   What I think of as a "query" in SQL is the part where you say `"SELECT * FROM tablename;"`. If you are asking about the differences between that and the "data" within the tables of an SQL database, then I think there's a massive difference. Data in an SQL database are rows, representing entities, and columns, representing attributes of those entities, all inside of tables (this is a simplification). Queries in SQL, are strings that have different keywords like `SELECT, WHERE, HAVING, GROUP BY` that instruct the system to do various operations to extract your data, but look nothing like how the data is stored.

   If instead, you are asking about the differences between the data and the *results* of a query - e.g. the rows you get back from the query you made - the data in those rows and the data in the tables looks very similar, in that they are both rows representing "entities" that have columns representing attributes.

4. What are the differences between the way the "data" and the "rules/queries" are represented in Prolog?

   Visually, the differences are very little. In Prolog, to store data, you store them as "facts". Facts looks like this:

```
person(suzie).
person(larry).
person(andy).

female(suzie).
male(larry).
male(andy).

parent(suzie, andy).
parent(larry, andy).
```

   They are made up of the name of the fact, and the data of the fact. They don't really have any meaning as they stand. There are the repeated names in the different facts, but nothing really connects them. Rules and queries are where these connections are made. They can be made up of several facts being compared with joins and unions.

Rules looks like this:

```
mother(X, Y) :- parent(X, Y), female(X).
father(X, Y) :- parent(X, Y), male(X).
```

Queries look pretty similar, but do not have that beginning part followed by the `:-`. Rules kind of look like a way to store longer query signatures, and are very powerful. A query using these above rules could look like:

```
mother(suzie, X).
X = andy;
false.
```

This query asked for the person Suzie is a mother to. And it replied with `X = andy;` The false afterwards just indicated that there are no other people that Suzie is a mother to.

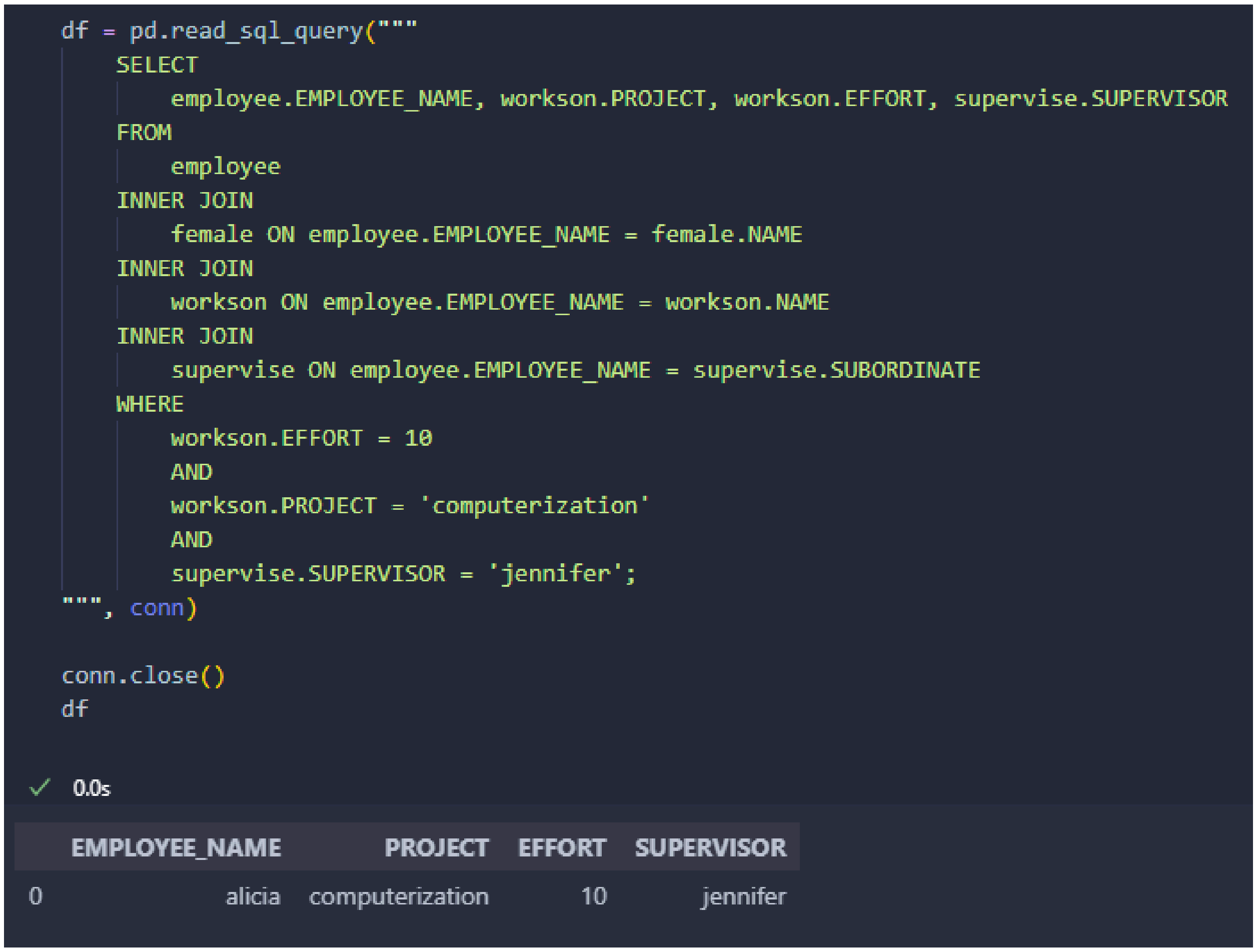
All in all, the way data and rules/queries look is very similar. Rules and queries are just operations on the data, similar to SQL queries, but they are displayed in a very similar way to the data itself.

Assignment Screenshots

Below are the screenshots of my programs working as intended.

Part 1: SQL

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.

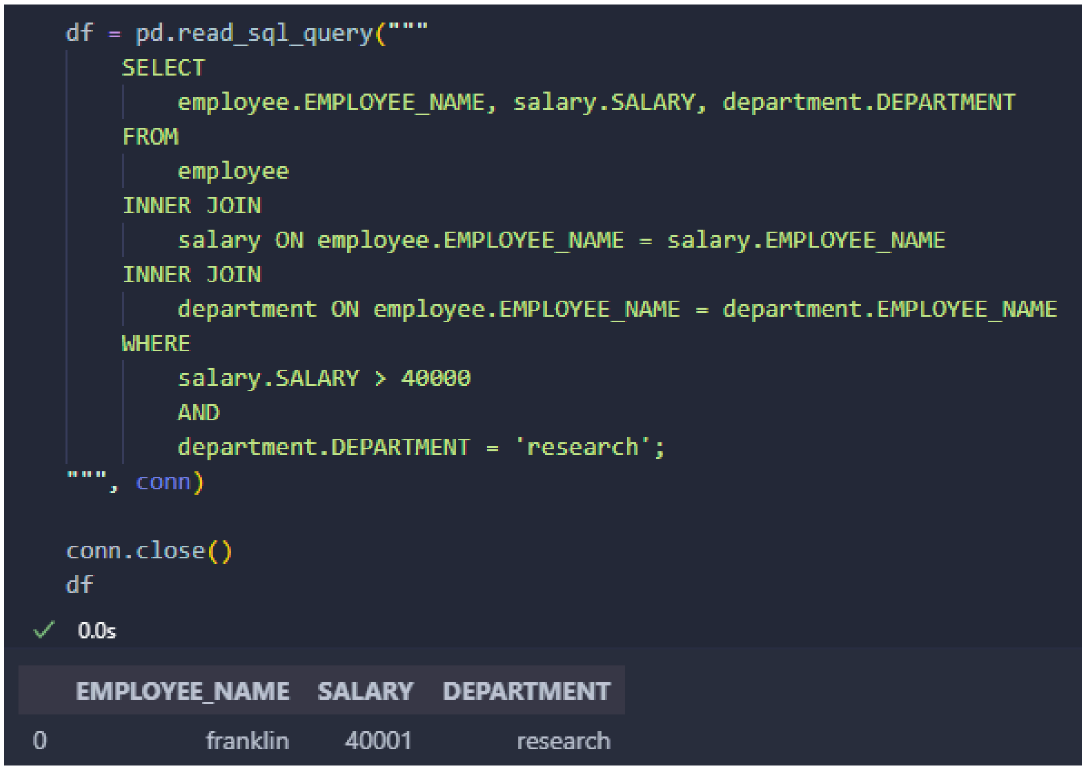


```
df = pd.read_sql_query("""
    SELECT
        employee.EMPLOYEE_NAME, workson.PROJECT, workson.EFFORT, supervise.SUPERVISOR
    FROM
        employee
    INNER JOIN
        female ON employee.EMPLOYEE_NAME = female.NAME
    INNER JOIN
        workson ON employee.EMPLOYEE_NAME = workson.NAME
    INNER JOIN
        supervise ON employee.EMPLOYEE_NAME = supervise.SUBORDINATE
    WHERE
        workson.EFFORT = 10
        AND
        workson.PROJECT = 'computerization'
        AND
        supervise.SUPERVISOR = 'jennifer';
""", conn)

conn.close()
df
```

```
✓ 0.0s
```

	EMPLOYEE_NAME	PROJECT	EFFORT	SUPERVISOR
0	alicia	computerization	10	jennifer

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

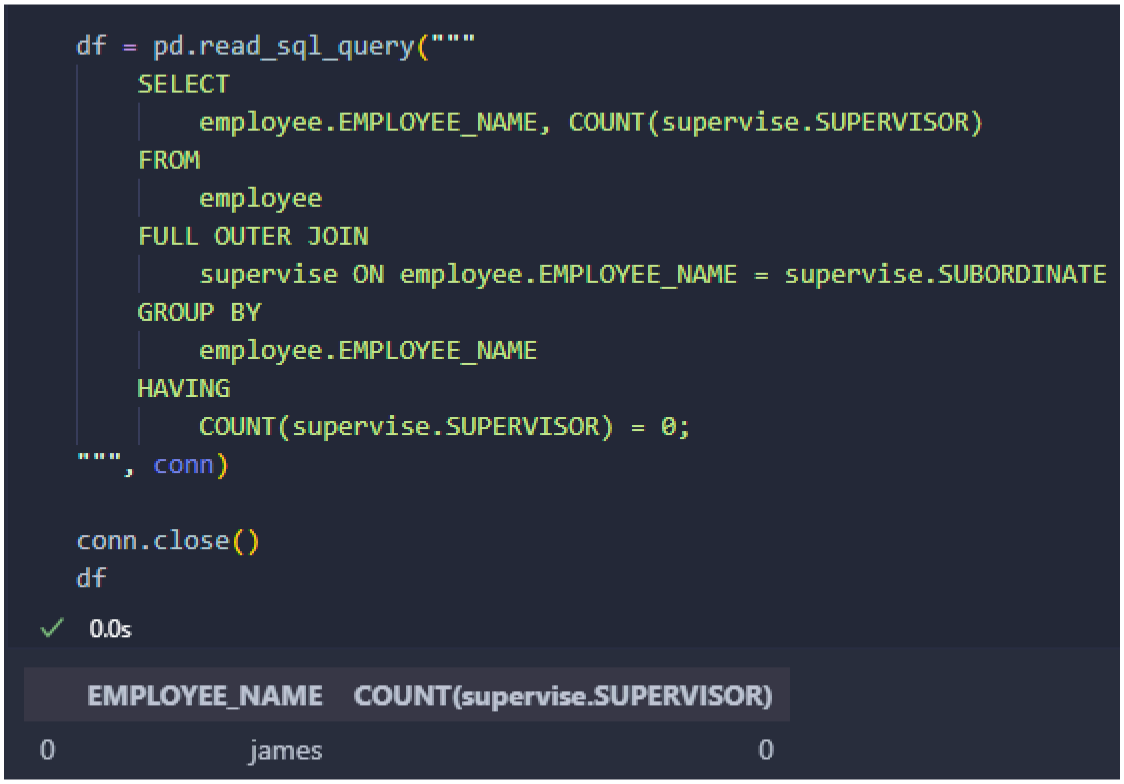


```
df = pd.read_sql_query("""
    SELECT
        employee.EMPLOYEE_NAME, salary.SALARY, department.DEPARTMENT
    FROM
        employee
    INNER JOIN
        salary ON employee.EMPLOYEE_NAME = salary.EMPLOYEE_NAME
    INNER JOIN
        department ON employee.EMPLOYEE_NAME = department.EMPLOYEE_NAME
    WHERE
        salary.SALARY > 40000
        AND
        department.DEPARTMENT = 'research';
""", conn)

conn.close()
df
```

✓ 0.0s

```
   EMPLOYEE_NAME  SALARY  DEPARTMENT
0       franklin   40001    research
```

Q3: Who is the supreme chief of this fictional company (aka the President)?

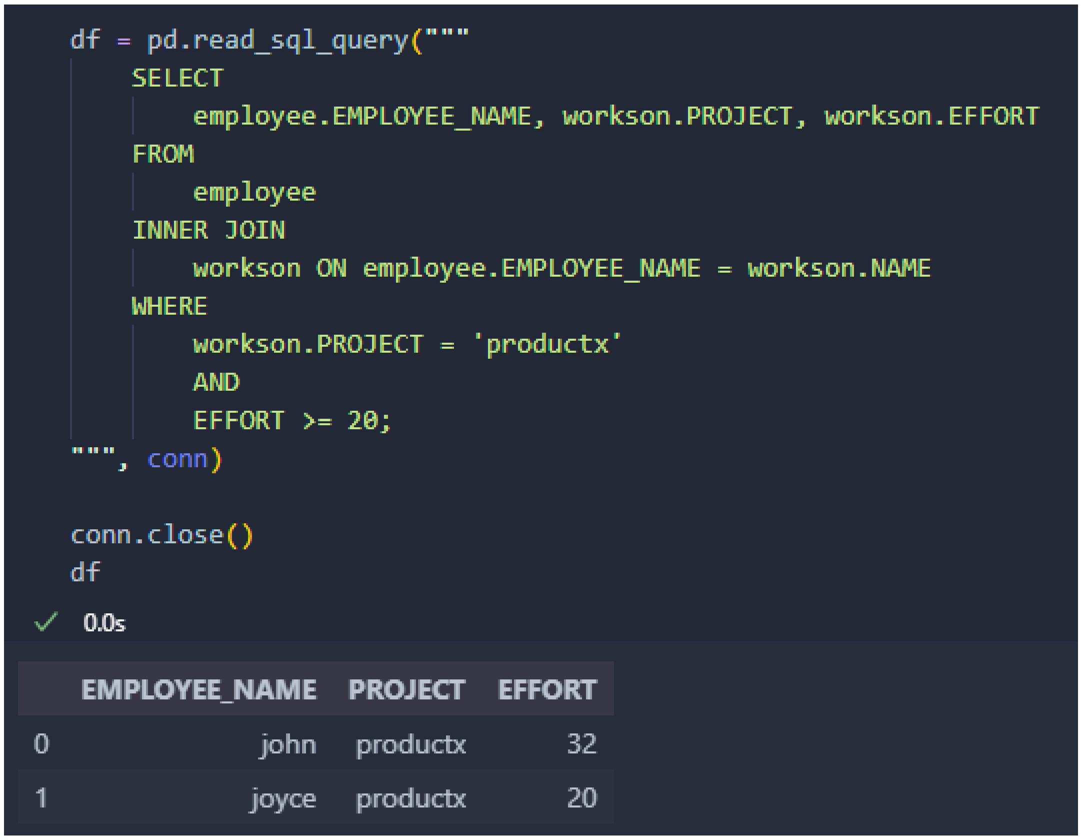


```
df = pd.read_sql_query("""
    SELECT
        employee.EMPLOYEE_NAME, COUNT(supervise.SUPERVISOR)
    FROM
        employee
    FULL OUTER JOIN
        supervise ON employee.EMPLOYEE_NAME = supervise.SUBORDINATE
    GROUP BY
        employee.EMPLOYEE_NAME
    HAVING
        COUNT(supervise.SUPERVISOR) = 0;
""", conn)

conn.close()
df
```

✓ 0.0s

```
   EMPLOYEE_NAME  COUNT(supervise.SUPERVISOR)
0          james                            0
```

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



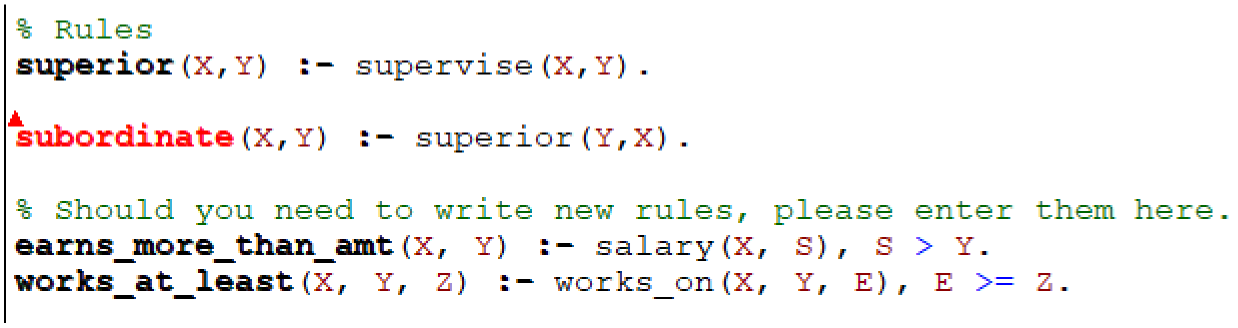
```
df = pd.read_sql_query("""
    SELECT
        employee.EMPLOYEE_NAME, workson.PROJECT, workson.EFFORT
    FROM
        employee
    INNER JOIN
        workson ON employee.EMPLOYEE_NAME = workson.NAME
    WHERE
        workson.PROJECT = 'productx'
        AND
        EFFORT >= 20;
""", conn)

conn.close()
df
```

✓ 0.0s

	EMPLOYEE_NAME	PROJECT	EFFORT
0	john	productx	32
1	joyce	productx	20

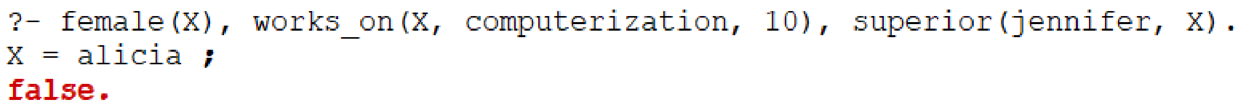
Part 2: Prolog

Here are all the rules that are in my prolog file:



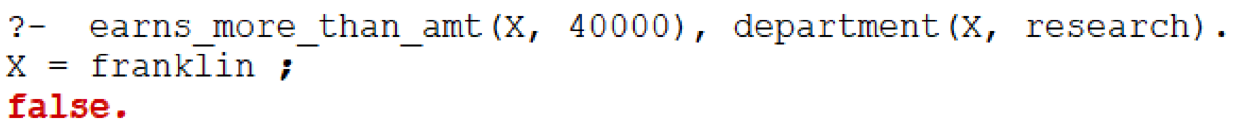
```
% Rules
superior(X,Y) :- supervise(X,Y).

subordinate(X,Y) :- superior(Y,X).

% Should you need to write new rules, please enter them here.
earns_more_than_amt(X, Y) :- salary(X, S), S > Y.
works_at_least(X, Y, Z) :- works_on(X, Y, E), E >= Z.
```

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.



```
?- female(X), works_on(X, computerization, 10), superior(jennifer, X).
X = alicia ;
false.
```

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



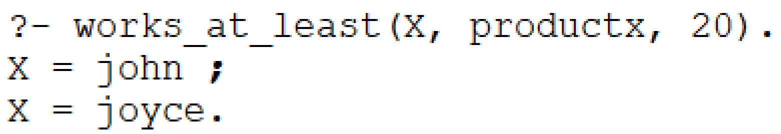
```
?-  earns_more_than_amt(X, 40000), department(X, research).
X = franklin ;
false.
```

Q3: Who is the supreme chief of this fictional company (aka the President)?

I could not figure this out on Prolog for the life of me. The way I solved it in SQL was using the `COUNT` function, but since (to my knowledge) I can't do that here, I had no idea how to find out whether someone did not have a supervisor to use in the query.

I also considered using the salaries to determine who was president. The highest salary should be the president's, right? But I wasn't able to figure out a query to use that either.

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



```
?- works_at_least(X, productx, 20).
X = john ;
X = joyce.
```