COMP-4522 ASSIGNMENT 2

IYAN VELJI
ALDO ORTIZ

Mount Royal University | Dr. Orestes Appel | March 16, 2024

INTRODUCTION

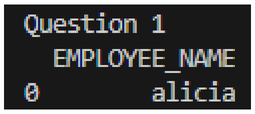
In the rapidly evolving space of Information Technology (IT), it remains imperative to understand the different kinds of programming languages in the field of database management such as Prolog and SQL.

Embarking on a scholarly endeavor, both Aldo Ortiz and Iyan Velji dive into the intracacies of SQL and Prolog navigating through employee records and filtering by gender, project involvement, and supervision.

We believe this assignment not only serves as an excellent comparative analysis of SQL and Prolog's approaches to data and query handling but also reflects a diverse understanding of computing systems.

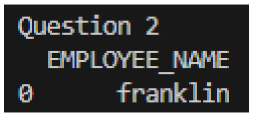
PART I: PYTHON AND SQL

For our first query we got the result of Alicia as the employee that is female, works in “computerization” with an effort of “10” hours per week, and has Jennifer as their supervisor.



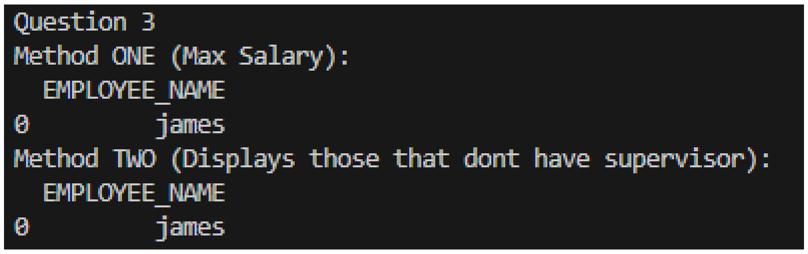
```
Question 1
  EMPLOYEE_NAME
0        alicia
```

Our second query returned Franklin as the employee making over $40,000 a year while working in the researcfh department.



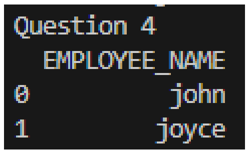
```
Question 2
  EMPLOYEE_NAME
0      franklin
```

To find the supreme chief we used two methods. The first method found the employee with the maximum salary. The second method displayed employees without a supervisor that still supervised someone. Both methods return James as the supreme chief.



```
Question 3
Method ONE (Max Salary):
  EMPLOYEE_NAME
0         james
Method TWO (Displays those that dont have supervisor):
  EMPLOYEE_NAME
0         james
```

Finally, we used a SELECT statement to search for the employees who work on productx with an effort of 20 or more hours a week. The answer is John and Joyce.

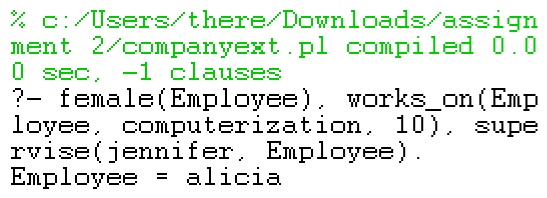


```
Question 4
  EMPLOYEE_NAME
0          john
1         joyce
```

PART II: PROLOG

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.

To solve this, we used the “,” symbol to combine the female, project, and superivsor facts using AND logic. The result returned by SWI-Prolog is Alicia.

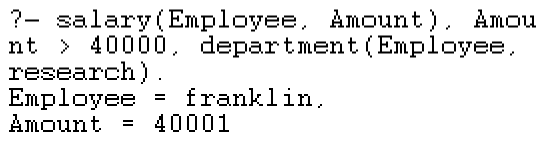


```
% c:/Users/there/Downloads/assign
ment 2/companyext.pl compiled 0.0
0 sec, -1 clauses
?- female(Employee), works_on(Emp
loyee, computerization, 10), supe
rvise(jennifer, Employee).
Employee = alicia
```

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

For this question, we used the query “salary(Employee, Amount), Amount > 40000, department(Employee, research).” integrating multiple conditions with the AND operator. A salary is searched for with an amount greater than $40,000 using the Amount variable. The employee is also matched to the research department. In the end, we get a result for the Employee variable that matches our criteria.

The answer is revealed to be Franklin.



```
?- salary(Employee, Amount), Amou
nt > 40000, department(Employee,
research).
Employee = franklin,
Amount = 40001
```

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

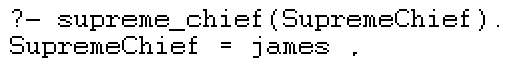
To answer this, we needed to make a new rule as follows:

```
supreme_chief(SupremeChief) :-
supervise(SupremeChief, _), \+
(supervise(_, SupremeChief)).
```

The idea here is that the SupremeChief would supervise people but would not be superevised by anyone themselves.

For the query we simply use the head supreme_chief(SupremeChief).

The answer returned is James.



```
?- supreme_chief(SupremeChief).
SupremeChief = james .
```

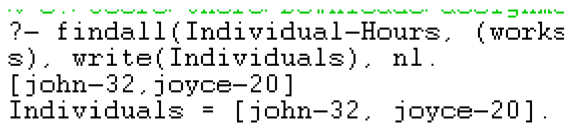
Q4: Who are the individuals that work on project "productx" with an *effort* of 20 or more hours a week?

Finally, to answer this question we queried the works_on fact with:

```
findall(Individual-Hours,
(works_on(Individual, productx,
Hours), Hours >= 20),
Individuals),
write(Individuals), nl.
```

This returns an answer to the Individual variable for someone who has worked on productx for over 20 hours as defined by the Hours variable.

The answer returned is John and Joyce.



```
?- findall(Individual-Hours, (works
s), write(Individuals), nl.
[john-32,joyce-20]
Individuals = [john-32, joyce-20].
```

REFLECTION

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

The most salient point between SQL and Prolog is that SQL is fundamentally relational while Prolog is based on first-order logic. Prolog and SQL are common in that they are both declarative languages with SQL using the DBMS to execute the query and Prolog using the Prolog Interpreter to execute the code.

The differences lie in how SQL and Prolog is written due to the nature of relational databases and deductive databases. The SQL queries focus on key relational operations like “JOIN”, “SELECT”, and “SELECT MAX(Salary)” to extract data from the CSV tables. In Prolog, one of the main differences was having to declare a new rule to obtain the supreme chief of the organization. This was not something that had to be done in SQL. The queries in Prolog then try to match the rules to the facts.

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

No, in SQL the logic is implicit within the query itself whereas in Prolog it is much more explicit with logical relationships clearly defined. This is beneficial for hierarchical and non-linear queries.

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

Data in SQL is stored in tables that are highly structured. This includes constraints like primary keys, foreign keys, indexes, columns, and data types. In our code, CSV files were imported into an SQLite database with each file representing a table corresponding to entities like employees and projects.

The queries in SQL aim to manipulate these tables using operations like SELECT, JOIN, FROM, and WHERE to filter and aggregate data based on specific criteria. Additionally, the queries in SQL operate within a strictly defined schema relying on integrity constraints and table relationships as part of the Entity-Relationship model. Queries in SQL use a declarative syntax specifying the data that is needed without needing to specify how to retrieve it. In addition, join operations allow SQL to combine tables based on related columns as demonstrated in our queries.

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

In Prolog, data is represented as facts which are fundamental assertions about entites and relationships. For example, “employee(john).” adds the employee John to the database. This allows for a flexible approach to data modelling that is appropriate for irregular or hierarchical data.

Prolog uses rules to infer information from facts. For example, supreme_chief(X) :- employee(X), not(supervise(_,X)). defines a rule to find the supreme chief by stating that it is an employee who is not supervised by anyone. Queries can be posed to the system in SWI-Prolog to check the validity of statements based on the existing facts and rules within the closed universe. This process can involve complex pattern matching and recursive search, leveraging the logical inference engine of Prolog.