# Assignment 2 – SQL & Data Frames and Deductive Databases

Matthew Lindstrom, Spencer Reid, Justin Savenko

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

   The most noticeable point between the commonalities and differences of queries formulated in SQL versus those expressed in Prolog lies in their fundamental approach to querying. SQL and Prolog are used to extract information, but their methodologies and purposes differ drastically. SQL queries are structured to interact with relational databases, focusing on data retrieval, manipulation, and management using a set of predefined commands tailored for working with structured data. In contrast, Prolog queries are formulated based on logical rules and relations, emphasizing logical inference and reasoning to derive answers. While SQL queries operate within the framework of tables and predefined schemas, Prolog queries operate within a logical knowledge base defined by facts and rules.

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

   The logic followed when writing queries in SQL and Prolog was slightly different due to the inherent nature of each language. When writing SQL queries, the logic often revolved around data manipulation and retrieval based on a predefined schema and relations. In contrast, Prolog queries involved more logical reasoning and inference based on rules and facts defined within the knowledge base. While writing queries in both languages required logical thinking, the thought process varied due to the distinct paradigms they adhere to. While SQL focuses on relation algebra and set theory, Prolog follows a declarative and logic-based approach.

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

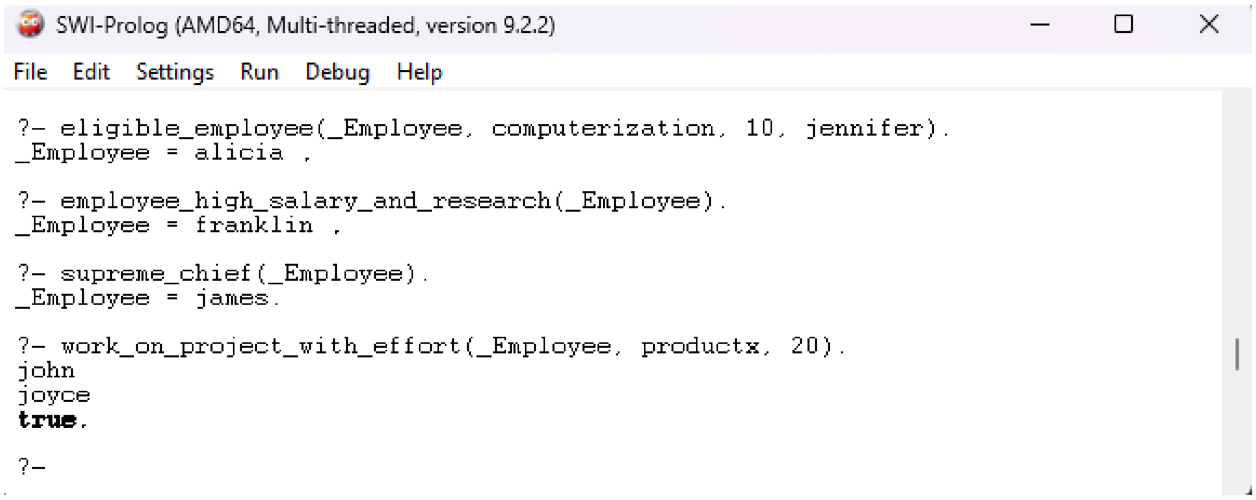
   The differences between data representation and queries in SQL are notable due to SQL's relational model. In SQL, data is stored in typically tables consisting of rows and columns, adhering to a predefined schema. Queries in SQL are expressed through the use of SQL statements like SELECT, INSERT, UPDATE, and DELETE, where conditions and clauses are applied to filter, manipulate, and retrieve from these tables. Data representation is structured and tabular, facilitating efficient storage and retrieval operations.

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

   In Prolog, the representation of data and rules/queries differs significantly from SQL primarily due to its logic-based paradigm. Data in Prolog is represented as a collection of facts and rules defining relations and properties among entities. Facts represent concrete information, while rules establish the logical relationships and conditions. Queries in Prolog are formulated as logical goals or queries that seek answers based on the available facts and rules within the knowledge base. Unlike SQL, which operates on structured tables, Prolog operates within a logical knowledge base, where data and rules are interlinked to facilitate logical inference and reasoning.

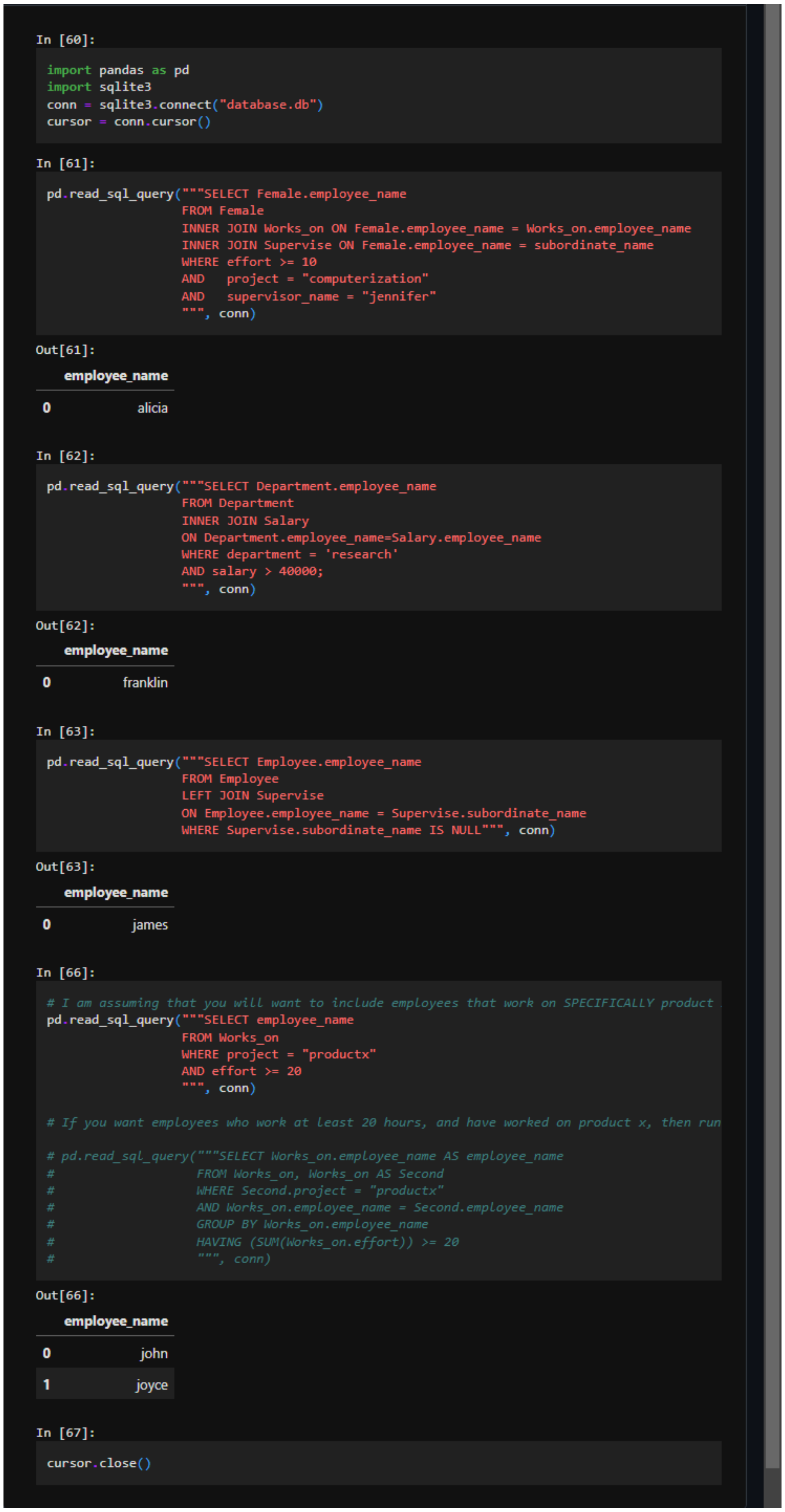
Assignment 2 – SQL & Data Frames and Deductive Databases
Matthew Lindstrom, Spencer Reid, Justin Savenko

Screenshots


SWI-Prolog (AMD64, Multi-threaded, version 9.2.2)
File Edit Settings Run Debug Help

```
?- eligible_employee(_Employee, computerization, 10, jennifer).
_Employee = alicia ,

?- employee_high_salary_and_research(_Employee).
_Employee = franklin ,

?- supreme_chief(_Employee).
_Employee = james.

?- work_on_project_with_effort(_Employee, productx, 20).
john
joyce
true.

?-
```

Assignment 2 – SQL & Data Frames and Deductive Databases
Matthew Lindstrom, Spencer Reid, Justin Savenko



In [60]:

```
import pandas as pd
import sqlite3
conn = sqlite3.connect("database.db")
cursor = conn.cursor()
```

In [61]:

```
pd.read_sql_query("""SELECT Female.employee_name
                  FROM Female
                  INNER JOIN Works_on ON Female.employee_name = Works_on.employee_name
                  INNER JOIN Supervise ON Female.employee_name = subordinate_name
                  WHERE effort >= 10
                  AND   project = "computerization"
                  AND   supervisor_name = "jennifer"
                  """, conn)
```

Out[61]:

```
   employee_name
0         alicia
```

In [62]:

```
pd.read_sql_query("""SELECT Department.employee_name
                  FROM Department
                  INNER JOIN Salary
                  ON Department.employee_name=Salary.employee_name
                  WHERE department = 'research'
                  AND salary > 40000;
                  """, conn)
```

Out[62]:

```
   employee_name
0       franklin
```

In [63]:

```
pd.read_sql_query("""SELECT Employee.employee_name
                  FROM Employee
                  LEFT JOIN Supervise
                  ON Employee.employee_name = Supervise.subordinate_name
                  WHERE Supervise.subordinate_name IS NULL""", conn)
```

Out[63]:

```
   employee_name
0          james
```

In [66]:

```
# I am assuming that you will want to include employees that work on SPECIFICALLY product
pd.read_sql_query("""SELECT employee_name
                  FROM Works_on
                  WHERE project = "productx"
                  AND effort >= 20
                  """, conn)

# If you want employees who work at least 20 hours, and have worked on product x, then run

# pd.read_sql_query("""SELECT Works_on.employee_name AS employee_name
#                     FROM Works_on, Works_on AS Second
#                     WHERE Second.project = "productx"
#                     AND Works_on.employee_name = Second.employee_name
#                     GROUP BY Works_on.employee_name
#                     HAVING (SUM(Works_on.effort)) >= 20
#                     """, conn)
```

Out[66]:

```
   employee_name
0           john
1          joyce
```

In [67]:

```
cursor.close()
```