Internship Report

**TOPIC**

**"Translation of Natural Language Queries to SQL Queries towards Building a Natural Language Interface to Database"**

**Submitted by:**Tiyasha Kundu  
B.Tech, 7th Semester  
Kalinga Institute of Industrial Technology, DU

**Duration:**[28/06/2024] to [28/07/2024] (1 Month)

**Under the Supervision of:**Sudeep Kumar Naskar

**Submitted to:**Jadavpur University

**Date of Submission:**[24/07/2021]

**Contact Information:**[[tiyashakundu3@gmail.com](mailto:tiyashakundu3@gmail.com) / 9330107685]

**Abstract:**

Project on developing an NL2SQL (Natural Language to SQL) system leveraging NLP (Natural Language Processing) to translate user-friendly natural language questions into structured SQL queries that can be executed against a database for an NLID (Natural Language Interface to Database) system. The system processes user queries, extracts relevant information, and generates structured SQL statements. The methodology includes tokenization, part-of-speech tagging, and syntactic parsing to accurately map natural language inputs to SQL queries. The key findings demonstrate the system’s ability to effectively translate a range of queries into SQL, improving accessibility and efficiency in database interactions. The implications of this work extend to simplifying database management for users with limited technical expertise and enhancing the overall user experience in querying databases.

**INTRODUCTION:**

The project titled "Translation of Natural Language Queries to SQL Queries towards Building a Natural Language Interface to Database" focuses on bridging the gap between non-technical users and database management systems. Traditionally, retrieving data from databases requires knowledge of SQL, which can be a barrier for many users. By developing a Natural Language to SQL (NL2SQL) system, this project aims to translate user-friendly, natural language queries into SQL statements. Leveraging Natural Language Processing (NLP) techniques such as tokenization, part-of-speech tagging, and syntactic parsing, the system enables more accessible, intuitive, and efficient database interactions. The goal is to simplify the database querying process and enhance the user experience, particularly for those with limited technical expertise.

**Model:**

**Input**

(nl questions)

**Text Processing**

Tokenization, stemming, lemmatization, pos\_tagging and punctuation removal

**Mapping**

Result Map to Synonym\_words of predefined examples

**Corpus**

Predefine question sets

**Text Processing**

Tokenization, stemming, lemmatization,punctuation removal,pos\_tagging, synonyms handling

**Mapping**

Map the synonyms with the SQL keywords

**Output**

SQL query Generated

(*Diagrammatic representation of the main idea of the project*)

**Methodology:**

**1 Text Processing of Corpus**

Step 1.1: we created a SQL table

**ID\_NO FIRST\_NAME LAST\_NAME AGE SALARY**

**------------ ------------------------------------ ------------------------------------ ---------- --------------**

**101 Anata Sarkar 28 55000**

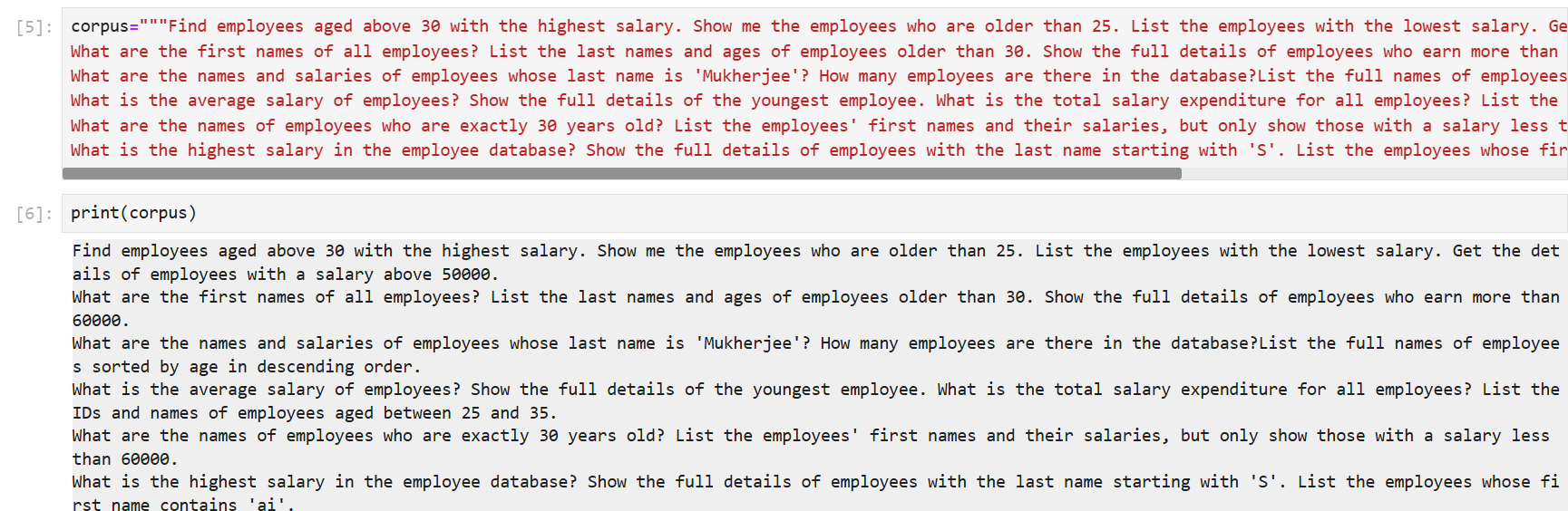
**102 Asutosh Mukherjee 34 63000**

**103 Rada Sengupto 25 45000**

**104 Bulai Chandra 45 80000**

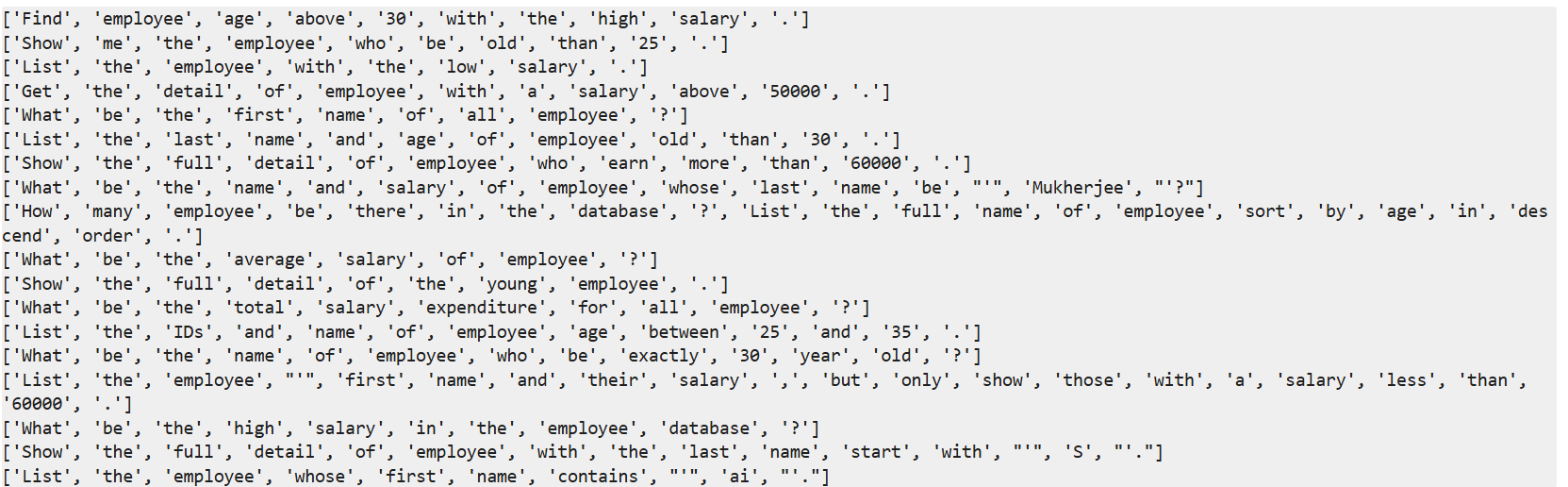
**105 Chaitali Mukherjee 30 57000**

Step 1.2: We created a corpus containing certain questions regarding the table.



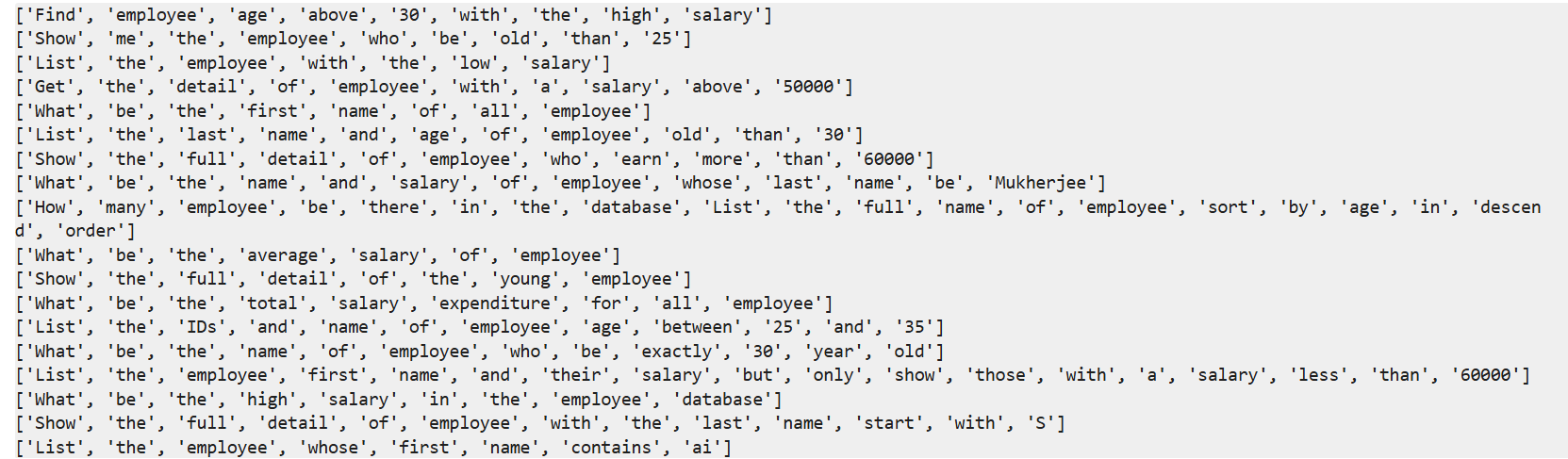
(*figure 1.1*)

Step 1.3: I tokenised the corpus. Tokenisation is important as it breaks down text into manageable units, such as words or phrases, which are essential for processing and analyzing text data enabling more precise text manipulation. Here the tokens are in the form of words and the punctuations are also separated.



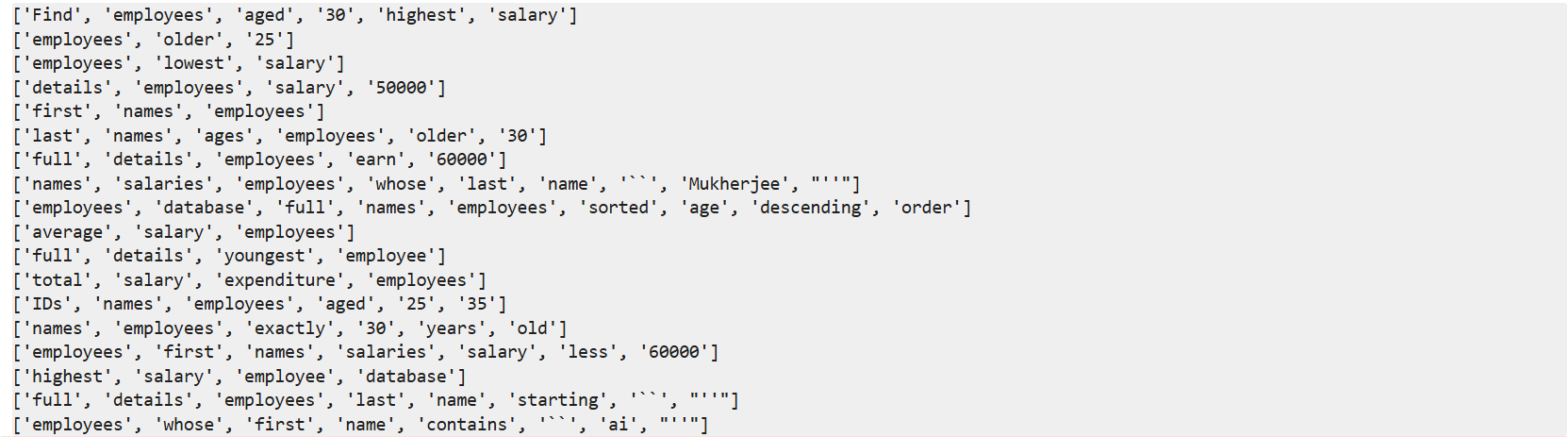
(*figure 1.2*)

Step 1.4: Lammetization is performed on the above token. It is important to lemmatize the tokens as it reduces words to their base or root form, ensuring that each of the different forms of a word is treated as a single item thereby enhancing text consistency and improving the accuracy of NLP tasks. Here I have also removed the punctuation as it increases readability and manipulation of tokens.



(*figure 1.3*)

Step 1.5: Here I also tried to remove the stopwords but after the removal of stop words it looked like this.



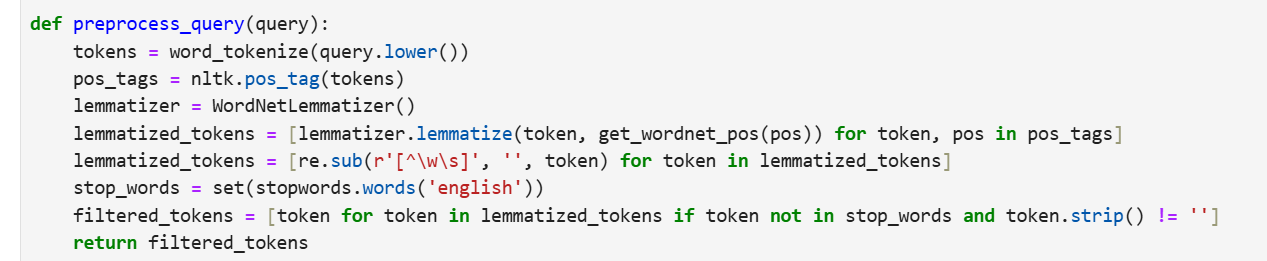
(figure 1.4)

But the problem is if we use stopword removal then a few important words will also be removed. As we can see in the 1st example of ‘figure 1.3’ & ‘figure 1.4’, ['Find', 'employees', 'aged', '30', 'highest', 'salary'] and ['Find', 'employee', 'age', 'above', '30', 'with', 'the', 'high', 'salary'] words like ‘above’ have been removed. And ‘age’ changed to ‘aged’, ‘employee’ to ‘employees’ and for the rest, the synonyms are removed e.g ‘find’ or synonyms of ‘find’ like ‘get’, ‘list’, ‘show’, from the rest is removed. These changes will give us a wrong analysis leading to a wrong SQL query.

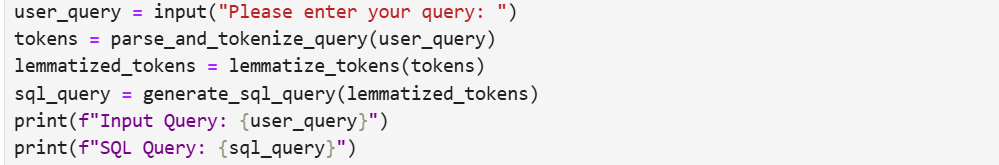
Step 6: The POS tagger model is used for both the corpus as well as nl sentence as the input. It is used to identify and label the grammatical parts of speech in a text, such as nouns, verbs, adjectives, etc helping in understanding the syntactic structure and meaning of sentences (here as tokens), improving the accuracy of various NLP tasks. For the corpus, lemmatization is done POS based and for the query, it is done while processing the query.

**2 Text Processing of NL inpute-based sentences for user-based approach**

Step 2.1: Here we also tokenised, lemmatized, Pos tagged and did all the necessary processing



(*figure 2.1*)



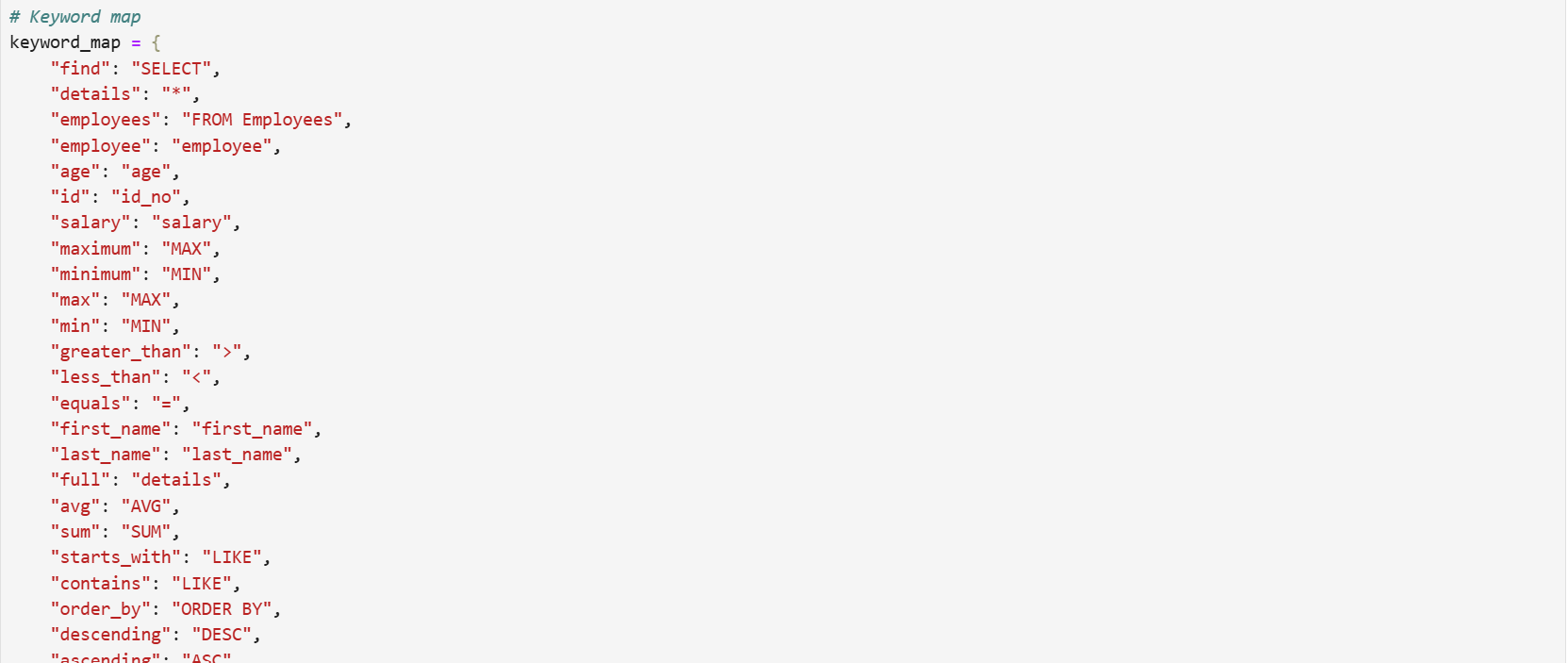
(*figure 2.2*)

**3 Mapping**

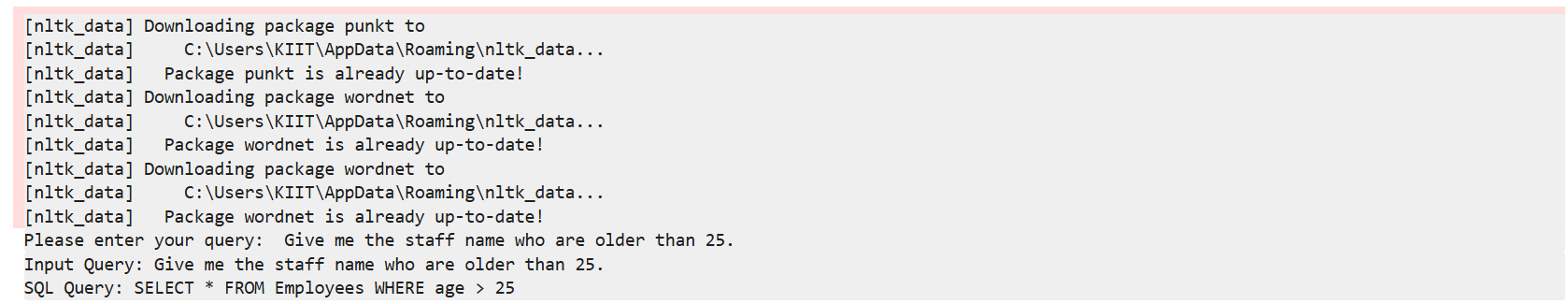
Step 3.1: At first I manually mapped all the synonyms as well as the SQL keywords.



(*figure 3.1*)



(*figure 3.2*)

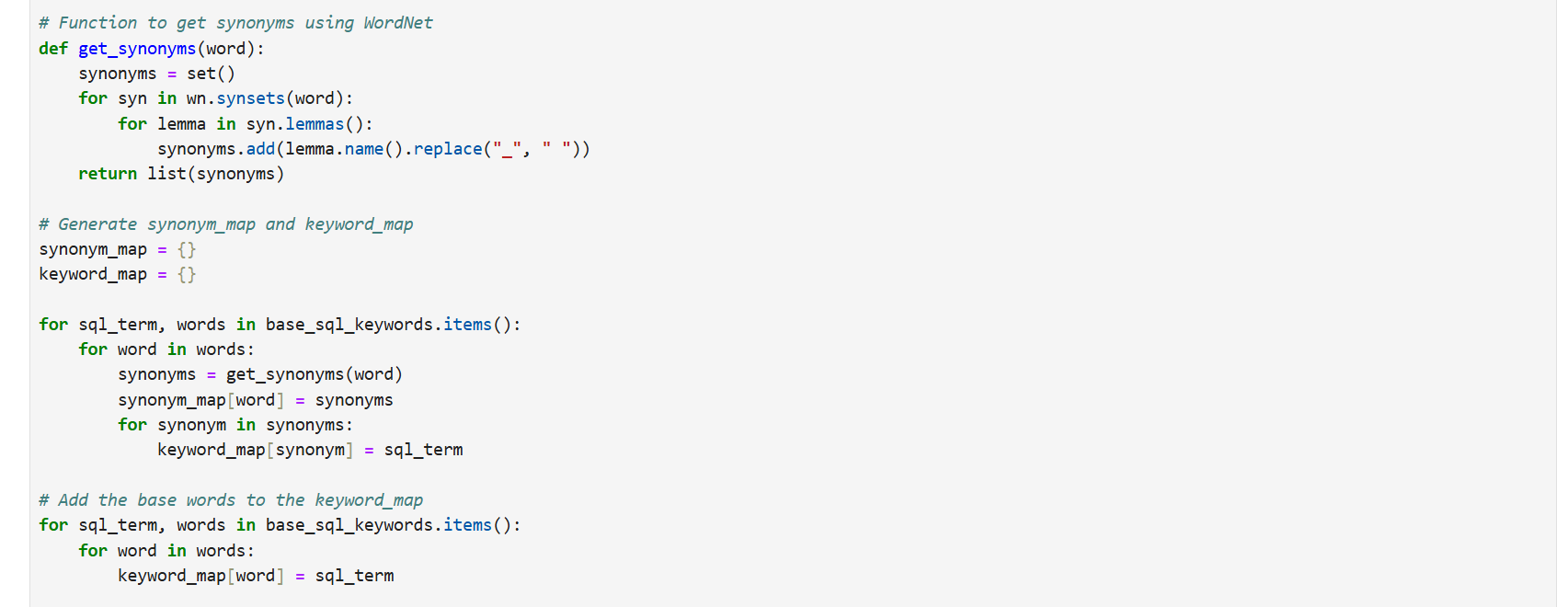


(*figure 3.3*)

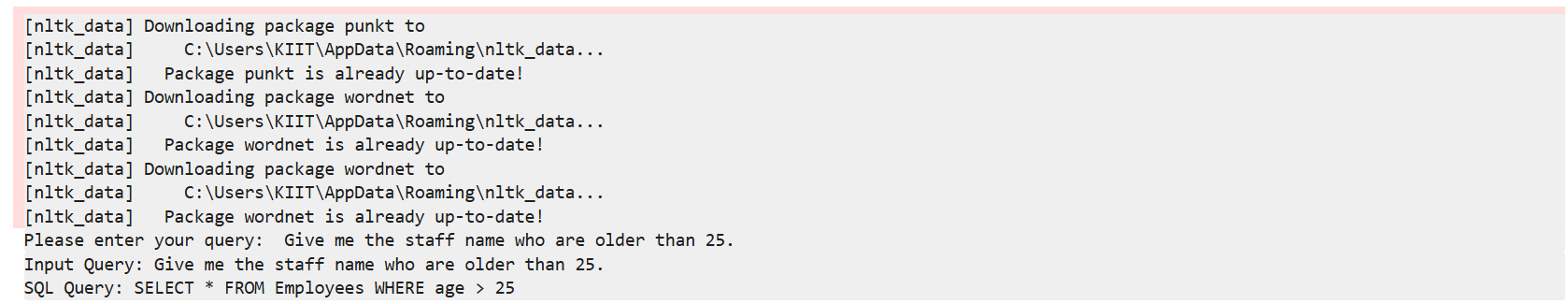
Which was working fine but automated mapping reduces the time, effort, and potential for human error involved in manual mapping, thereby enabling scalable, consistent, and efficient handling of large datasets and complex queries, enhancing overall productivity and accuracy in data processing tasks.



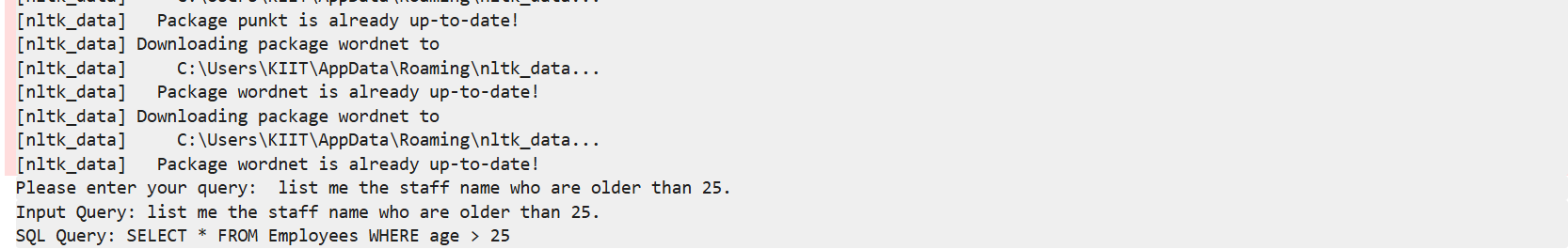
(*figure 3.4*)



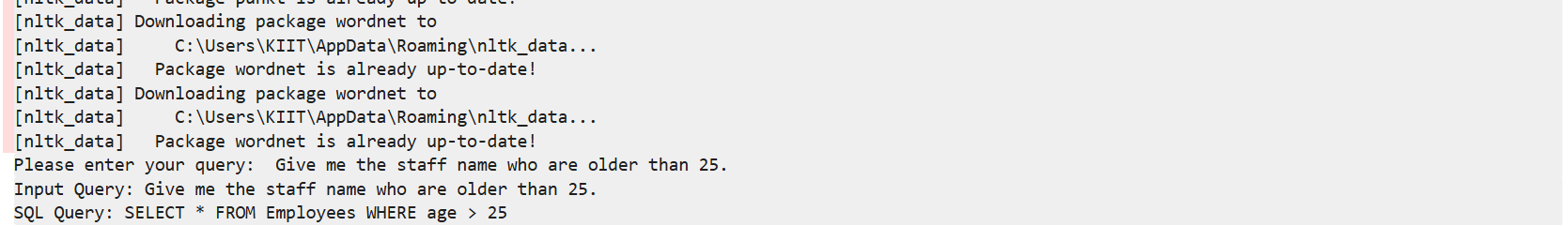
(*figure 3.5*)



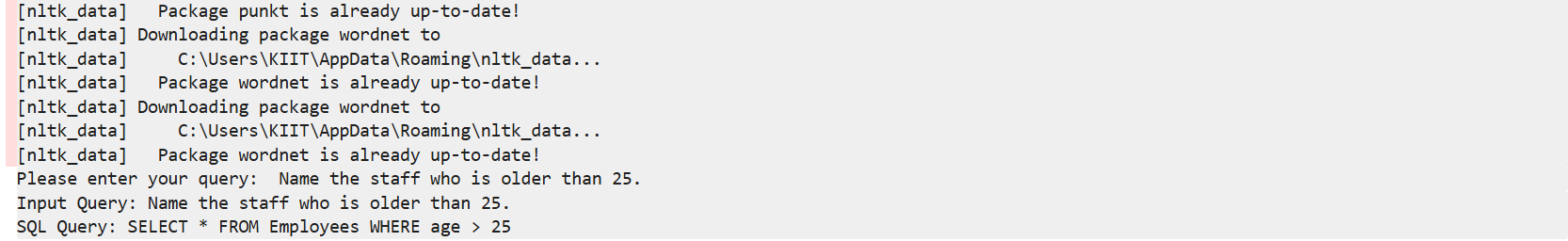
(*figure 3.6*)



(*figure 3.7*)



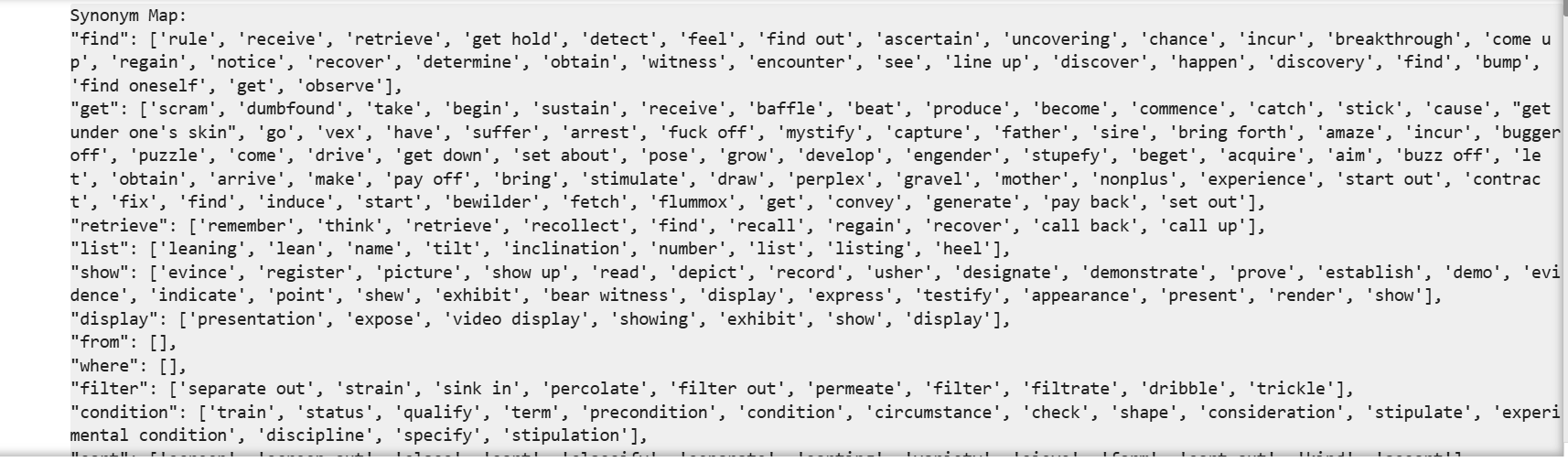
(*figure 3.8*)



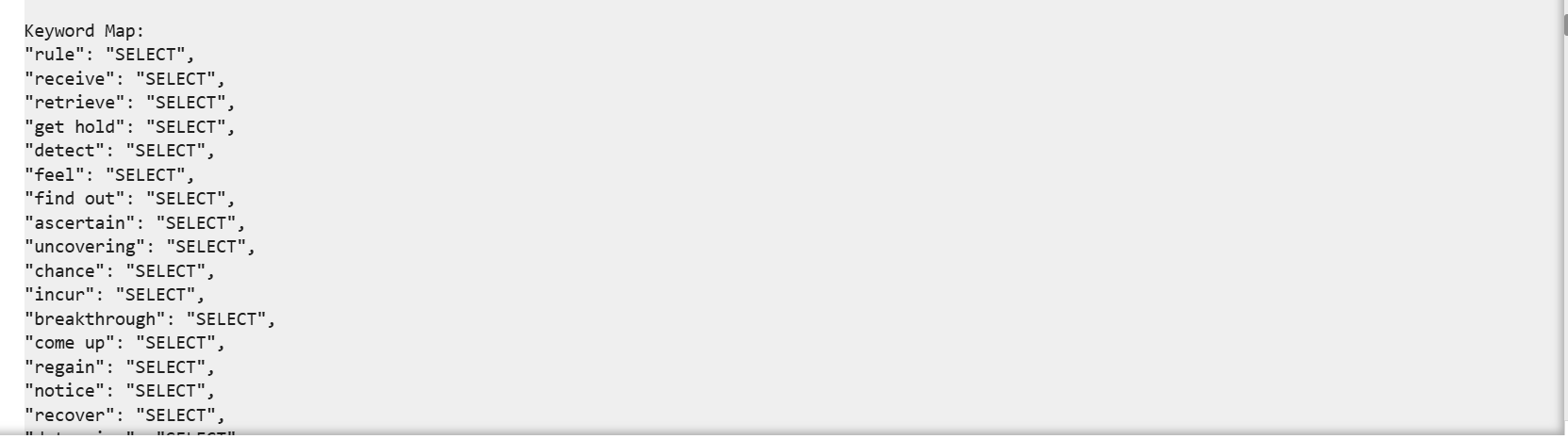
(*figure 3.9*)

This shows that the Mapping is working fine for a variety of different NL sentences as input.

This was a problem as when done manually for some examples it won't recognize the query but the hurdle was passed using Wordnet. This the how the Mapping is working-



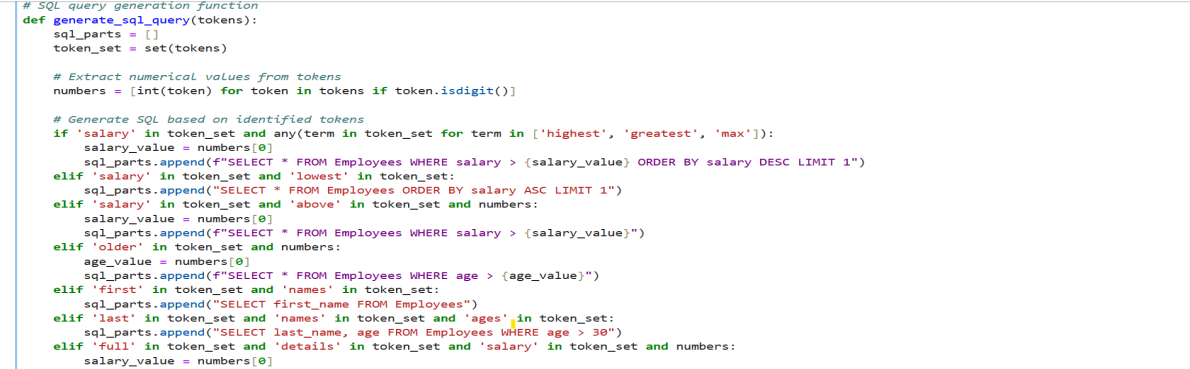
(*figure 3.10*)



(*figure 3.11*)

**4 SQL output**

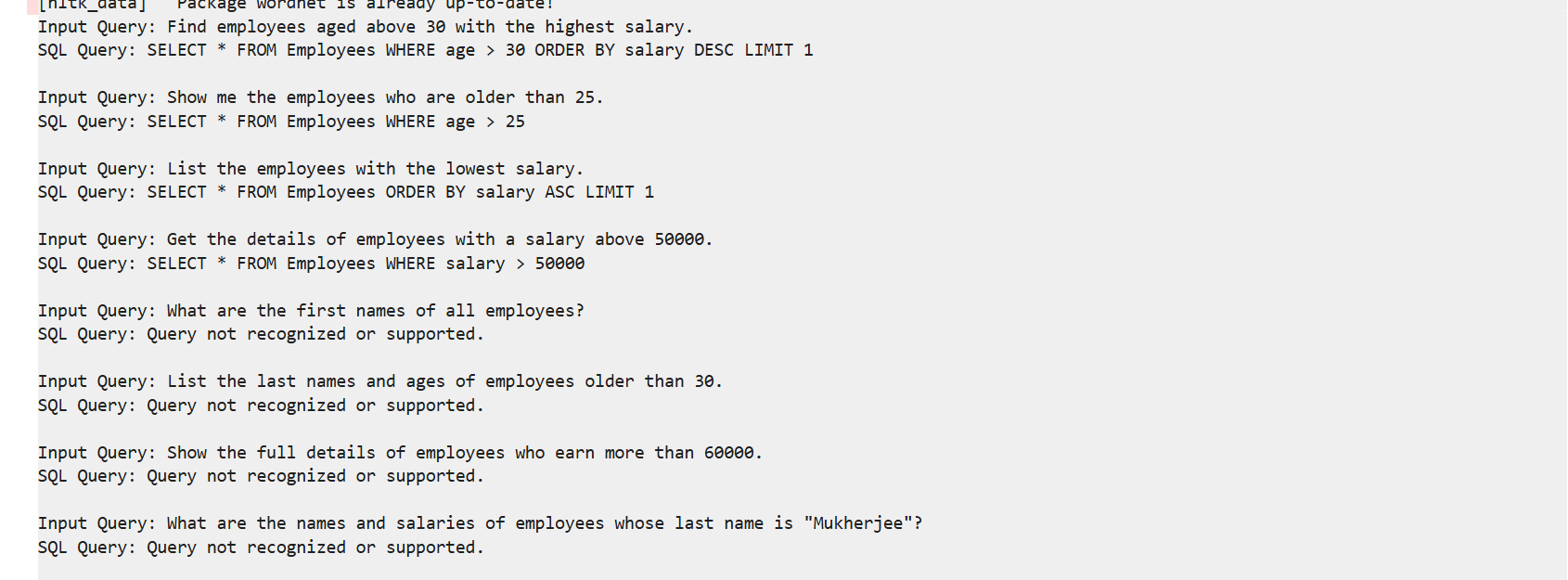
Step 4.1: we will write a function to generate SQL connecting to all the mappings of the tokens



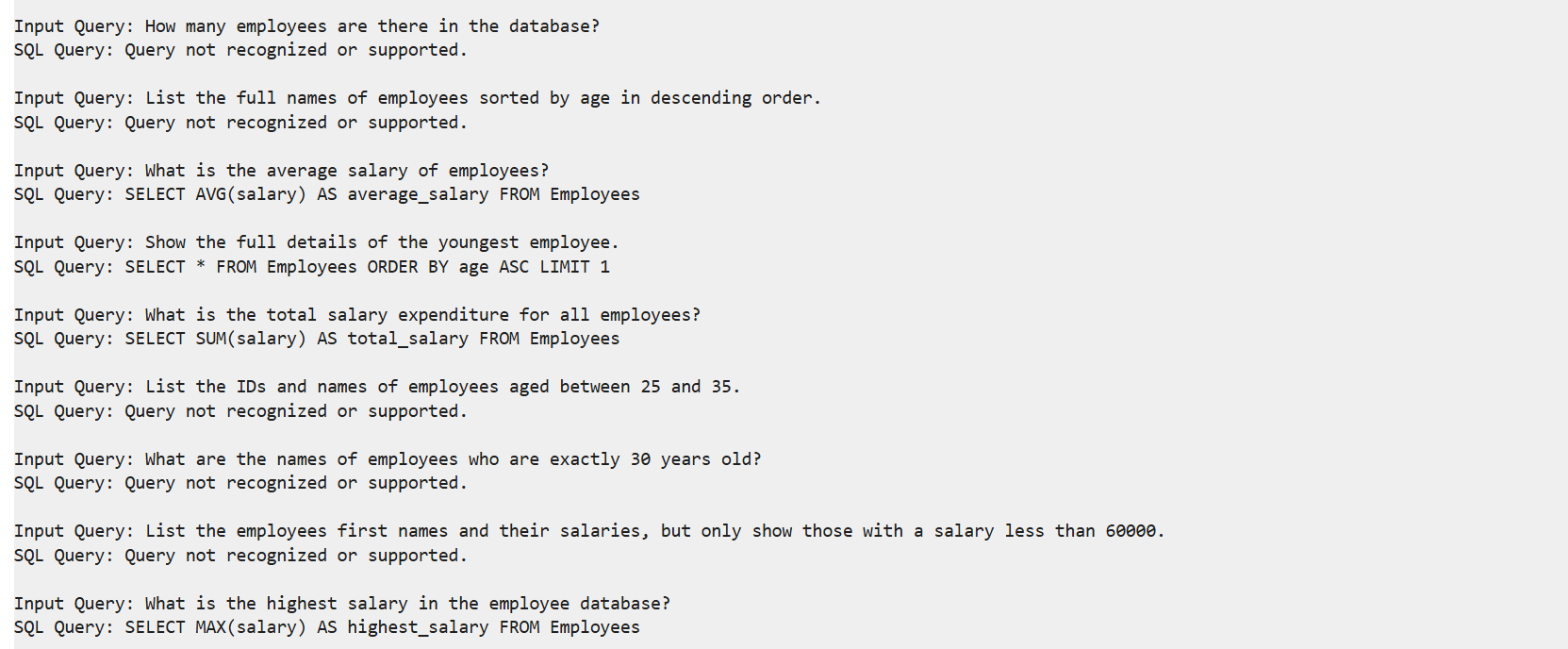
(*figure 4.1*)

I have generated user based as well as example based outputs. For example based for 18 queries 10 queries are running successfully with any varied combination of natural language sentences.

EXAMPLE BASED:

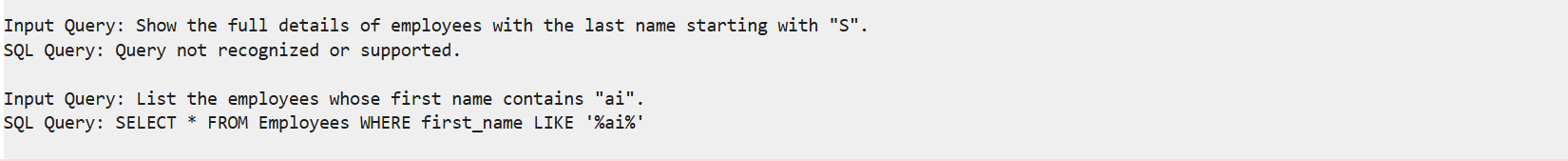


(*figure 4.2*)

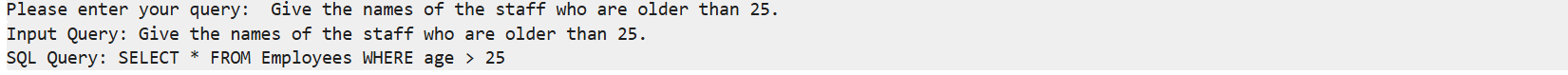


(*figure 4.3*)

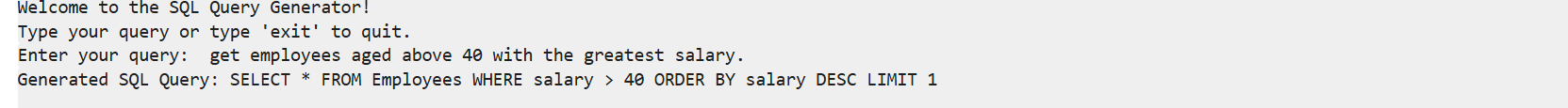
USER APPROACH:



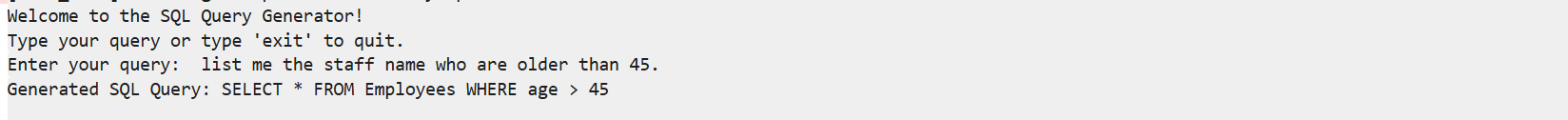
(*figure 4.4*)



(*figure 4.5*)



(*figure 4.6*)



(*figure 4.7*)

### **Internship Experience**

#### **1 Tasks and Responsibilities**

During my internship, I was primarily responsible for developing and enhancing an NL2SQL (Natural Language to SQL) system. My key responsibilities included:

* **Designing and Implementing the NL2SQL System:**

Developed the core functionality to translate natural language queries into structured SQL commands.

Implemented NLP techniques to parse user inputs and map them to SQL queries.

* **Query Processing and Translation:**

Worked on processing user queries to extract relevant entities and intent.

Created algorithms to generate accurate SQL queries from natural language inputs.

* **Testing and Validation:**

Conducted extensive testing to ensure the system handled various types of queries effectively.

Validated query results against expected outputs to confirm accuracy.

#### **2 Projects or Assignments**

1. **NL2SQL System Development:**

Focused on creating a system that translates natural language queries into SQL. Key features included handling synonyms and numerical comparisons.

1. **Example Query Testing:**

Tested various queries to assess the system’s performance in handling different scenarios, such as age comparisons and synonym mappings.

1. **Performance Evaluation:**

Evaluated the system’s accuracy and efficiency by comparing the output of natural language queries with expected SQL results.

#### 

#### **3 Tools and Technologies Used**

1. **Programming Languages:**

Python for developing the NLP algorithms and SQL query generation.

1. **NLP Libraries and Frameworks:**

**NLTK (Natural Language Toolkit):** For tokenization, lemmatization, part-of-speech tagging, and parsing.

**spaCy:** For named entity recognition and syntactic parsing.

1. **Database Systems:**

**SQL Database:** For testing and executing the generated SQL queries.

1. **Development Environment:**

**Jupyter Notebook:** For prototyping and testing NLP models.

#### **4 Challenges Faced and Solutions**

* **Handling Synonyms:**

**Challenge:** Mapping synonyms corectly and variations in user queries to correct SQL keywords.

**Solution:** Developed a comprehensive synonym dictionary and implemented semantic similarity measures to improve accuracy. Later automated it using lemma in sys.

* **Numerical Comparisons:**

**Challenge:** Parsing and handling complex numerical comparisons (e.g., age ranges, salaries).

**Solution:** Designed flexible parsing rules to recognize and process multiple numerical conditions in queries using token sets.

* **Ensuring Accurate Query Mapping:**

**Challenge:** Ensuring that generated SQL queries accurately reflect user intent and synonyms are properly mapped.

**Solution:** Implemented rigorous testing and validation procedures to verify the correctness of SQL translations for varieties of synonyms and numeric ranges.

* **System Performance:**

**Challenge:** Balancing accuracy with performance to handle a wide range of queries efficiently.

**Solution:** Optimized algorithms, models and query generation function processes to improve response time and accuracy.

### **Skills Gained During the Internship**

### **Natural Language Processing (NLP):**

### Enhanced skills in applying NLP techniques, including tokenization, part-of-speech tagging, and named entity recognition.

### Gained experience in developing algorithms to process and translate natural language inputs into structured data.

### **SQL and Database Management:**

### Improved proficiency in SQL, including query formulation and execution.

### Gained practical experience in working with databases and understanding query optimization.

### 

### **Programming and Software Development:**

### Advanced coding skills in Python, particularly for implementing NLP and database interactions.

### Developed expertise in using development tools such as Jupyter Notebook and VS Code for coding and debugging.

### **Problem-Solving and Debugging:**

### Refined problem-solving abilities by addressing complex challenges in query translation and system performance.

### Enhanced debugging skills through rigorous testing and validation processes.

### **Project Management and Documentation:**

### Gained experience in managing a technical project, including planning, execution, and documentation.

### **Key Findings**

* **Query Success Rate:**

Out of 18 tested queries, 10 were successfully executed, demonstrating a medium performance rate. This indicates the effectiveness of the NL2SQL system in handling certain queries efficiently.

* **Handling Synonyms:**

The system effectively maps synonyms, allowing for flexible user input few major queries. For instance, variations in phrasing for age comparisons (e.g., "age above 25" versus "age above 20 or 35 or 45 ") are accurately processed, showcasing robust synonym handling. Synonyms for ‘find’, ‘show’, ‘get’ etc were covered. Even a wrong sentence phrase or minute spelling mistakes of certain words are also interpreted correctly and the query is correct

* **SQL Keyword Mapping:**

The queries that were executed successfully were well mapped to SQL keywords, ensuring accurate translation from natural language to SQL. This alignment contributes to efficient and correct query execution.

### **Conclusion**

The NL2SQL system demonstrates a significant proficiency in translating natural language sentences into SQL queries, with a success rate of 55% across diverse queries, the system effectively manages synonyms and numerical comparisons, enhancing its usability and flexibility. The successful mapping of SQL keywords ensures that the system executes queries accurately and efficiently. This indicates that the NL2SQL system is almost well-equipped to facilitate user interactions with databases, making complex queries more accessible and manageable for users with varying levels of technical expertise with future upgradation and training. Future work may focus on expanding the range of supported queries and further refining the system’s ability to handle more complex and varied inputs as well as establishing the connection between the database and Python for a better NLID system.

**Disadvantage**

* This will not work on complex queries.
* Sql generation is right but mapping of few of the not working queries needs more attention.
* This is not connected to the database so no direct communication is possible.
* A few synonyms of the example queues like ‘what, who, where and how’ might make even an easy query, difficult to interpret.

**Necessary Improvements**

* Need to make it more dynamic for a variety of complex queries
* Mapping can be done using Json file to dynamically allocate synonyms which will be specifically oriented for particular databases and would give better results with generic training.
* Creating an NLID system connection between Python, database and SQL is important which could be updated in future.
* The inclusion of more synonyms and coding SQL generation function more robustly and debugging might help find the possible hurdles and incompetencies in generating few SQL queries.
* Needs training and upgradation for the future success of 90% of SQL generation
* Langchain can be incorporated to generate an LLM system to convert NLP2SQL and make it more auto-generative even with a variety of languages like Hindi, Bengali, Telegu, Arabic etc.
* SQL documentation in the form of pdf might be used to auto-generate the SQL keywords by using PyPDF2 and perform text processing and data chunking to extract embedded keywords for SQL scripting

### **Acknowledgements**

I would like to express my sincere gratitude to my supervisor, Dr. Sudeep Kumar Naskar Sir, for his invaluable guidance and support throughout the internship. My thanks also extend to Mr Atanu Sir at Jadavpur University for his collaboration and assistance. Additionally, I am grateful to my professors and mentors at Kalinga Institute of Industrial Technology for their encouragement and academic support.