

MATH 2191 Applied Research Project Natural Language Query to Structured Query Language using Large Language Models (P000129DA)

Industry: RMIT University Data and Analytics
Industry Supervisor: Sean Lee, Director Advanced Analytics

No	Name of Team Member	Contribution
1	Enson Sim Zhan-Kai (s3952024)	50%, Executive Summary, Introduction, Literature Review, Conclusion
2	Beau Krieg (s3272094)	50%, Methodology, Results, Discussion, Appendix

Student Contributions Statement:

Beau Krieg designed methodology and analysed the results. Enson Sim led the writing for the introduction and the literature review. All students contributed critically to the report and gave final approval for submission.

Date of Submission: 29/10/2023

RMIT University
2023

Disclaimer

We agree and acknowledge that:

1. We have read and understood the Declaration and Statement of Authorship.
2. If we do not agree to the Declaration and Statement of Authorship in this context, the assessment outcome is not valid for assessment purposes and our academic integrity is void.

We declare the following to be our work, unless otherwise referenced, as defined by the University's policy on plagiarism. All the work taken from various sources have been referenced as per the required referencing standards thus we declare that we have ensured our academic integrity.

Acknowledgement

We would like to thank:

Sean Lee, our industry supervisor, for providing us with this project and taking the time to have regular meetings with us to provide his feedback on our work.

Dr. Yan Wang, our course lecturer, for assisting us by answering our course-related questions and enquiries for the assignments.

We would also like to express gratitude to Sona Taheri, our academic supervisor, for being available to be a mediator between us and the industry supervisor in the potential event that a serious issue arises that cannot be resolved between our two parties.

Table of Contents

Executive Summary	5
Introduction	6
Project Context	6
Research Objectives	7
Literature Review	8
An Overview of Natural Language Interfaces to Databases	8
Non-Neural Network Approaches.....	8
Machine Learning/Neural Network Approaches.....	9
Transformer Architecture.....	11
Solution Proposal	13
Conceptual Design	13
Model Discovery	15
Model Deployment and Features.....	18
RESDSQL.....	18
DIN-SQL.....	19
Model Evaluation Methodology	20
Results.....	22
Discussion	24
DIN-SQL Approach	24
RESDSQL approach	25
Comparative Discussion	26
Conclusion	27
Recommendation	27
Further study	27
References	29
Appendix A: Test Design	31
Web-based Front Interface for Display to the User	31
Queries used for Model Evaluation	32
Appendix B: Output from NLIDB evaluation scripts	34
RESD-SQL Evaluation Testing	34
DIN-SQL Evaluation Testing	45

Executive Summary

The rapid advancement of Large Language Models (LLMs) in the recent year has created much discussion about their potential usage in a variety of fields. Of particular interest to us is the ability of LLMs to generate outputs that are contextually correct with only minor errors based on inputs that are in the form of natural everyday language as commands. This has prompted a question of whether it would be possible for a person with limited computer programming knowledge to use LLMs to overcome their deficiency and thus bypass having to go through a dedicated IT department for non-essential inquiries.

The purpose of this project as given by RMIT University is to assess LLMs' capability to convert natural language questions to Structured Query Language (SQL) statements that can retrieve the correct data from relational database tables using the appropriate SQL schema. Firstly, we went through an extensive literature review to gain a deeper understanding of the problem and to identify models that were mature enough to quickly deploy in our project. We identified two fundamentally different solutions through our research that can answer our main objective. The first approach was to fine-tune an open-source model (RESDSQL + RoBERTa) and the second approach was to utilize prompt engineering to direct a Generative Pre-trained Transformer (DIN-SQL + GPT-4). Both models used Python for deployment.

We then experimented with the two approaches and recorded our results in this report. Using university level database concepts tutorial questions for our model evaluation and benchmarking, we found that DIN-SQL (matched 23/27 queries) significantly outperformed RESDSQL (matched 10/27 queries). The second approach was found to be simpler, faster, less computationally intensive and highly adaptable. We recommend that DIN-SQL be used in the future of this project if it is ever continued.

Introduction

Project Context

The effective utilization of data is central to the success of any enterprise today. Businesses and organisations rely on a diverse range of databases to maintain their core functions. However, accessing and manipulating this data often poses significant challenges, particularly for business professionals who lack the technical expertise required to undertake the complex querying and processing of this data.

A common scenario in many organizations is that only a few professional database administrators possess the capability to interact with databases efficiently. As a result, most users find themselves constrained by a limited set of pre-programmed, standard queries or reports, which have been carefully crafted by database admins. These fixed query structures, although useful for routine operations, will continue to lag behind evolving business needs. This lack of flexibility and adaptability in querying databases hinders business professionals from accessing real-time data insights that can aid in timely and informed decision making.

Recent advances in Large Language Models (LLMs) and Natural Language Processing (NLP) present an opportunity to bridge the gap between business professionals and database admins by enabling natural language interaction with databases. These technologies make it feasible for business professionals to pose complex queries or undertake complex database operations using everyday language, and subsequently develop meaningful and context-aware responses. This innovative approach eliminates the dependency on pre-programmed query structure and empowers users to interact with databases more dynamically and intuitively.

Wouldn't it be great if database users could interact and gain meaningful insights from data using conversational language? With this outcome in mind, data analytics professionals from RMIT University proposed a project to develop a natural language interface for relational databases using LLMs. This report details the development of a Natural Language Interface to Database (NLIDB) tool to meet their business requirements.

Research Objectives

The aim of this project is to enable business professionals at RMIT, with limited database knowledge, to use natural language to intuitively extract meaningful data from relational databases. These business professionals are assumed to be well versed in the content of such databases, however, lack the technical skills to query them using Structured Query Language (SQL) statements.

To keep within the scope of our project, we will focus on the below research questions that are pertinent to our problem statement:

1. What is the most advanced current Natural Language Interfaces to Databases (NLIDB) and what are their strengths and weaknesses?
2. What publicly available databases are available and suitable for the development and testing of NLIDB?
3. How might recent advances in LLMs be leveraged to increase the accessibility of complex datasets?

Literature Review

An Overview of Natural Language Interfaces to Databases

The idea of querying databases using Natural Languages (NL) has been around since the 1970s and has led to a plethora of approaches and implementations of various complexity, capabilities and accuracies. Such interfaces have a primary aim, converting NL into formal Query Languages (QL) such as Structured Query Language (SQL). These capabilities have seen incremental improvement over time; however, it was the more recent advances in Machine Learning (ML) that has supercharged research and development in this area and seen a large uptick in the number of NLIDBs and an increase in their functionality. Modern NLIDBs can now be classified by the distinction of either being based on ML approaches (neural networks) or otherwise not.

Non-Neural Network Approaches

In their 2019 review paper, (Affolter et al., 2019) introduce and benchmark 24, non-ML NLI. These are categorised as follows:

1. **Keyword based systems** use look-up implementations to match query keywords against a data index. These systems can only answer simple queries and cannot answer aggregation queries. Example implementations include SODA, NLP-Reduce, Precis, QUICK, QUEST, SINA, Acqu
2. **Pattern-based systems** extend keyword-based systems to handle more keywords and some natural language patterns, for example, to enable aggregation queries though keywords such as 'by' or 'how many'. These systems can also be programmed to account for domain specific concepts,

such as 'great movie' or 'best performer.' Example implementations include: NLQ/A, QuestIO

3. **Parsing based systems** attempt to understand the grammatical structure of the input question, which can then be mapped to defined production rules to generate a query in the target query language. Examples include ATHENA, Querix, FREyA, BELA, USI Answers, NaLIR, BioSmart
4. **Grammar based systems** have a defined set of rules that restrict the type and form of questions that can be asked in NL; however, only formalised questions can be answered by the system. Example implementations include TR Discover, Ginseng, SQUALL, MEANS, AskNow, SPARKLIS, GFMed.

These non-ML based systems can perform very accurately on some queries, however, even the most complex are limited in that they: don't generalise well (i.e. they are domain or database specific); they are limited in the complexity of the queries that they can handle; and while they accept NL queries, the range of NL that they accept is limited, and depends on the specification of keywords that often must conform to particular grammatical structures. The point could be made that these systems are just more forgiving querying languages, rather than true NLDBs.

[Machine Learning/Neural Network Approaches](#)

ML based approaches require suitable training data to build a model that is capable of translating NL to formal QL. In their review paper, (Iacob et al., 2020) discuss various strategies for training databases, including:

- **Fully supervised**, where the NLDB model is training on a dataset of paired NL statements and associated SQL queries.

- **Weakly supervised**, where the model is trained using only NL paired with correct answers (output of the query from the database).
- **User feedback**, where the user is used as a source of supervision to indicate the accuracy of the output or to help solve ambiguities.

For the case of fully supervised models, Iacob et al. (2020) identifies and summarise twelve, open-source text-to-SQL datasets for single turn interactions (i.e., no user feedback). Of these, nine are domain specific (i.e., cover only a single domain) and consist of only a single database. Of the three remaining multi-domain databases, WikiSQL (Zhong et al., 2017) is the largest, consisting of 80,654 queries of 26,251 databases, however is limited in that each database consists of a single table. The OTTA dataset consists of 3,792 queries of 5 databases (Deriu et al., 2020). Despite having many tables per database (12) of the multi-domain databases, it doesn't contain any nested SQL queries. The final dataset, the Spider dataset, consists of 10,181 queries, over 200 databases spanning 138 domains, with an average of 5 tables per database and 844 nested SQL queries (Yu et al., 2018). This dataset was designed to overcome limitations of other datasets when training semantic parsing (SP) models, in it covers multiple domains, and it has a higher number of SQL logic forms.

In addition to generating and publishing the Spider dataset, (Yu et al., 2018) introduce relevant evaluation metrics (component matching, exact matching and execution accuracy) and categorise questions according to their difficulty (easy, medium, hard and extra hard) based upon their complexity. This formal structure has enabled Yale University to host challenges to develop NLQ-to-SQL models, for which developers,

academics and other interested parties can build models and benchmark their performance on this dataset. The current leaderboard for this challenge can be found at <https://yale-lily.github.io/spider>.

Transformer Architecture

At the most basic level, LLMs are artificial neural networks which have been conceptualized and built upon since the 1960s. These neural networks utilize machine learning to train themselves in processing information with the aim to generate an output based on the given input. This output is analyzed and then the ML task is repeated with certain changes made to its algorithm. The new output is compared to the previous output and the procedure is looped successively until a satisfactory output is produced that matches the targeted output (Zhao, 2023).

The key breakthrough that led LLMs to its current spotlight in society was the introduction of transformer architecture in 2017. As proposed by Google researchers (Vaswani, 2017), transformers allowed ML to both require less time and reach the target output more accurately due to allowing memory to be stored virtually on the network itself using attention mechanisms. Hence, there is no longer a need to constantly adjust the ML algorithm manually as the stored memory was able to adjust it automatically based on the parameters set (number of loops to run, the changes to make if target does not match etc.).

Transformers utilize an encoder-decoder framework to perform the training process. The encoder part of the transformer allows for data to be submitted to use for training. The dataset is converted into tokens (the stored memory) by a byte pair encoding

tokenizer. The tokens are then processed to become semantically meaningful representations, for example a poem translated into code that can be understood by the neural network and allow for it to generate an output via inferencing. The decoder part then converts the tokenized output back into a humanly understandable form of text while keeping part of the generated output in its self-attention module.

The self-attention module also allows for rule storage, given by the user input, so that the LLM can “remember” how to show its output even if different commands are given to it (also known as prompt engineering). Through this method, a user can fine-tune the model by increasing the number of encoder-decoder layers for inferencing and increasing the weight of desirable tagged tokens to ensure that the wanted output is generated.

On the global stage, the LLMs which have been making massive disruptions are known as generative pre-trained transformers (GPT, from which the most popular LLM gained its namesake). The three largest LLMs that are available for either commercial or research purposes are GPT-4 developed by OpenAI, PaLM 2 developed by Google and LLaMA 2 developed by Meta (see Figure 1). All current commercial LLMs are decoder only because it is generally agreed that it would be too complex for a general user with little programming knowledge to make use of both functions. Also, the developers of these LLMs do not want outside data to contaminate the datasets which they use to train their product.

Interface (basically a text input) and demonstrates the components of the system that will need to be constructed. The tool will consist of:

1. A front end, the user interface, consisting of:
 - a. An Input Interface that will accept Natural Language Queries (NLQ)
 - b. A query pre-processor, that will combine the NLQ with the database schema, and transform it into a suitable sequence for the transformer layer (the back end).
 - c. A Display Interface, that will present results to the user (and make them available for export)
2. A data source, consisting of
 - a. The relational database that is being queried.
 - b. An interface to query the database using SQL (for the database managers).
3. A back end, consisting of the transformer module that will translate the NLQ to SQL. This back end may be hosted locally, or be cloud based and accessible via an application programming interface.

The development of the NLIDB tool will follow a software development process. The individual modules will be developed independently of each other, taking care of how the modules will connect with each other. The core capability of the tool is the transformer module, and a solution will be sourced from the open literature. The NLIDB development is summarised in the subsequent sections.

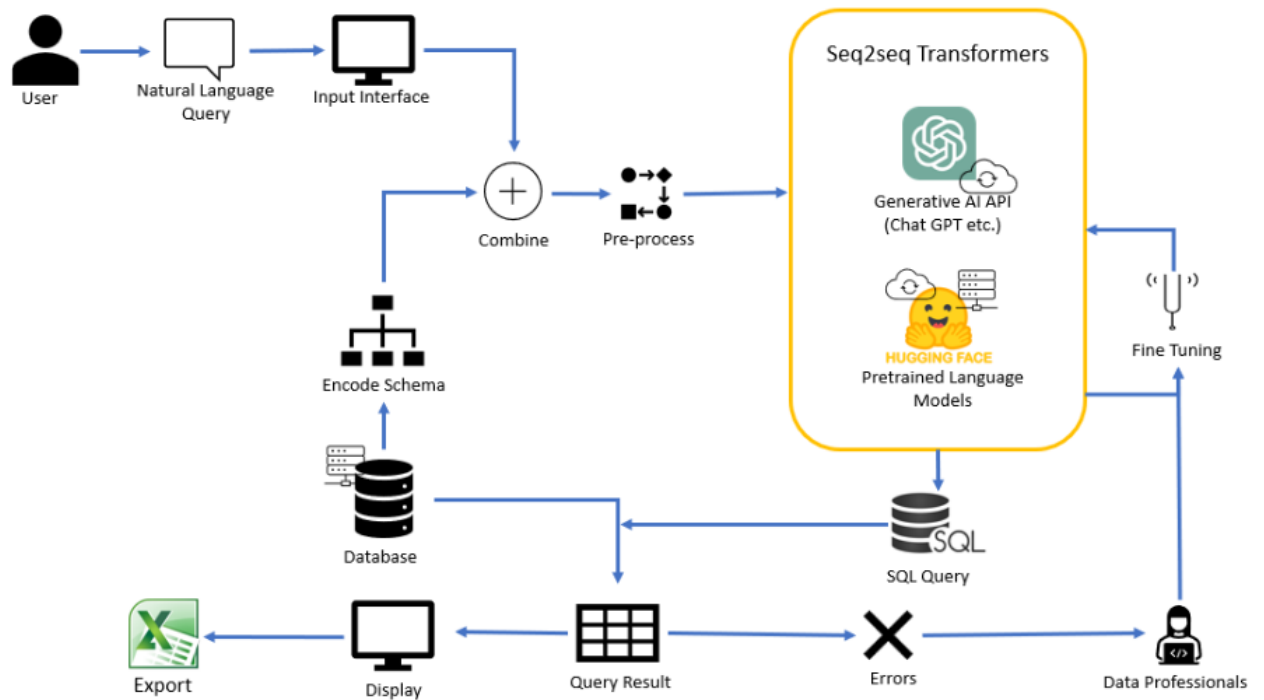


Figure 2: CogniQuery (our project team name) NLIDB program/system general outline

Model Discovery

The top solutions¹ on the Spider dataset were examined, and two solutions were down selected. A quick examination of approaches shows several different strategies and approaches for solving the machine learning tasks of semantic parsing and text-2-SQL conversion.

Problem decoupling

The task of text-to-SQL is divided into multiple, simpler tasks. Commonly, this involved separating the *schema linking* task, where required fields from the database are identified, from the *semantic-parsing* task, which seeks to understand how the various fields are meaningfully related to the NLQ.

¹ NOTE: This is a very active area of research. The down select of models occurred during the August/September 2023. At the time of writing this report, changes in the leaderboard had already occurred.

Chain-of-thought

Similar to problem decoupling, the chain-of-thought approach considers the structure of the problem that is posed to the large language model. For example, the DIN-SQL model prompts a LLM to first classify the NLQ-to-SQL conversion task according to its complexity, and then use solve the problem using different strategies based on that complexity (chain-of-thought approach).

Preprocessing

The database schemas were frequently pre-processed, as to as much as possible standardise the inputs to better fine-tune LLM and aid in the generalisation between databases. Preprocessing may also have been used to standardise feature names or replace feature names with more semantically meaningful names (e.g., *mvtitle* becomes *movie_title*).

Prompt engineering

Several approaches focused on developing broadly effective prompting templates that posed the NQL and solution strategy to the LLMs as to maximise the probability to providing a correct response (Zhao, 2023). This was typical of models that used the GPT style models.

Model fine-tuning

Approaches such as the RESDSQL approach (Li et al., 2023), sought to fine-tune transformer models as to maximise their accuracy at particular tasks. The RESDSQL authors trained two different models, one to undertake *schema linking*, and another to identify and populate the correct SQL patterns of keywords and database fields.

Self-Correction

The ML approaches to NLQ to SQL, while broadly effective, can be quite buggy. NL has more flexibility than a formal coding language, and so syntax mistakes are

common. Self-correction approaches seek to clean the output, either by calling the transformer models again to correct its own mistakes, or by post-processing the results to correct common errors.

At the time of writing this report, the top performing approaches could be largely separated by the primary approach they take to improving performance, i.e., the fine-tuning transformer models versus using prompt engineering approaches with GPT style models. The two best approaches for each class for these were the RESDSQL+ NATSQL and DIN-SQL (Pourreza & Rafiei, 2023) respectively. Both were considered for inclusion in the NLIDB tool, and their performance on the Spider dataset is summarised in the table below.

Table 1: Execution Accuracy on the Spider Development and Test sets, as reported in (Li et al., 2023) and (Pourreza & Rafiei, 2023) for the RESDSQL and DIN-SQL approaches. Exact Match (EM) refers to the predicted SQL query exactly matching the ‘gold’ SQL query described by the Spider dataset, whereas Exact set match (EX) accuracy refers to a matching result from the executed SQL queries.

Model	Dev Set (%)		Test Set (%)	
	EM	EX	EM	EX
RESDSQL-Base	71.7	77.9	-	-
RESDSQL-Base + NatSQL	74.1	80.2	-	-
RESDSQL-Large	75.8	80.1	-	-
RESDSQL-Large + NatSQL	76.7	81.9	-	-
RESDSQL-3B	78.0	81.8	-	-
RESDSQL-3B + NatSQL	80.5	84.1	72.0	79.9
DIN-SQL + GPT-4	-	-	60	85.3

Model Deployment and Features

The original published code for both the RESDSQL and DIN-SQL approaches utilised Python scripts to batch process the Spider dataset and evaluate the model effectiveness. To deploy these models, the original, published code was modified and incorporated into a custom Python class, the SQL database class which implemented the following methods:

Nlq – Natural Language Query function. Takes a NLQ as a string and returns the predicted SQL statement.

Sql – Structured query language function. Takes a SQL statement as a string, queries the connected database and returns the results.

RESDSQL

The Ranking Enhanced Schema Decomposition (RESDSQL) approach has the following features:

- Implemented in Python 3.8.5
- The schema-linking task is decoupled from the SQL generation task.
- Schema-linking is undertaken by a fine-tuned, RoBERTa transformer model, trained on the Spider dataset. This model ranks all database features (tables and columns) according to their perceived importance in answering the question.
- The semantic-parsing task is undertaken by constructing an empty SQL skeleton, consisting of the required keywords for the query, with empty slots of the database schema items. For example: 'SELECT ____ FROM ____ JOIN ____ ON ____'. A T5 transformer model (trained on the Spider dataset) was fine-tuned using the Spider dataset for this task.

- The SQL skeleton is populated with the schema links.
- Self-correction is undertaken in two passes:
 - Extensive post-processing of common mistakes is undertaken.
 - Since multiple potential (ranked) SQL statements are produced by the T5 transformer, these are tested by querying the actual database. The highest ranked executable SQL statement is returned.

When testing the RESDSQL model, Li et al. (2023) trained several T5 models. These transformer models are very large, with even the smallest (the T5-Base model) sized at approximately 870 MB. As such, only the Base version could be run on the test computer without running out of RAM (test computer had 8GB RAM). The implemented version from Table 1 is the RESDSQL-Base + NatSQL model.

DIN-SQL

DIN-SQL stands for Decomposed In-Context Learning of Text-to-SQL. This model/code consists of a series of prompt templates that are chained together to decompose the NLQ-to-SQL problem into a set of intermediate steps that follow a particular reasoning strategy. These flow as follows:

- The GPT-4 model is requested to undertake schema linking, to identify the key database features required to answer the question.
- Based on the schema links and the NLQ, the GPT-4 model is asked to classify the complexity of the query as:
 - *Easy* – single table queries, no joins or nesting required.
 - *Non-nested complex* – queries that require joins, but not sub-queries.
 - *Nested complex* – queries require joins, sub-queries and set operations.

- Different prompt templates are provided for each level of complexity. The schema links and NLQ are passed to the corresponding template, and the GPT-4 model called.
- A final prompt template requests the GPT-4 model to check and correct the SQL query generated in the previous step (self-correction).

Several modifications to the code needed to be made to get it operational. It is suspected that updates to the GPT-4 model, post publication of the original code, provided slightly different outputs. This made the functions that specifically extracted pertinent information from the outputs error prone. In addition, several other modifications were made to the prompt templates to encourage cleaner output for parsing, or to improve the efficiency of the SQL conversion when trying to match a string in the database by encouraging case insensitive matching.

Model Evaluation Methodology

Two testing scripts were developed to assess the effectiveness of each tool at completing two SQL tutorials from a postgraduate university database course. These tutorials (Sumanasekera, 2017) asked students to construct SQL queries to answer a series of NLQ about the Movies database. Unfortunately, this Movies database lacked logical field names. The authors modified this database to also create the CleanMovies database, where field names were replaced with more semantically meaningful names. E.g., MVTITLE → movie_title, NOMS → award_nominations etc. The CleanMovies database was used to assess the RESDSQL approach, while the original Movies database was used to assess the DINSQL (it was found to be just as effective on the original database).

Appendix A contains all the questions taken from the two tutorials. These were divided into three broad categories:

- **9 Basic queries** which referenced a single table and performed simple tasks such as filtering or sub-setting the data.
- **6 Advanced queries** which required advanced operations on a single table, including ordering, arithmetic operations (e.g., averaging, difference), time-based arithmetic operations and creating new fields. One query required broader inferencing when it requests dates be displayed in 'Australian standard format'.
- **12 Complex queries** which required multiple joins, operations and nested queries to solve. Broader inferencing is often required, such as understanding that 'directors who are still alive' are those that don't have the date of their death listed.

Success was considered if the output could be reasonably considered to accurately meet the intent of the NLQ and selected the correct data entries and minimally required columns, even if it didn't fully meet the exact tutorial solutions. For example, for the first question, '*What are the movies available in your collection?*', the target output selected all columns from the Movie table, however DIN-SQL selected only the movie titles. This response met a reasonable interpretation of the question.

Once classified as either Correct or Incorrect queries, the Incorrect queries were assessed qualitatively to classify the type of error based on commonalities in the errors.

Results

The DINSQL approaches successfully answered all Basic, Advanced and all but 4 of the Complex queries. One failure related to failing to exactly match a database entry when using the '=' operator; while the last was a minor failure where the results were wrongly limited to a single result when their multiple results in the set. The remaining failures were attributed to semantic errors. These arise when the precise wording of the questions was somewhat difficult to decode, and the model failed to predict joining nuances such as self-joins and outer joins verses regular joins.

The RESDSQL approach displayed much poorer performance, successfully matching target query results for 5 of the 9 basic questions, and 2 each of the 6 advanced, and 12 complex queries.

Amongst the Incorrect queries, the following error types were identified:

- **Equality** – These occurred when the query required exact matching, however the model could not correctly interpret the format of the match. For example, both models failed the NLQ: *'Display movies that are directed by Woody Allen'* as this name is encoded as 'Allen, Woody' in the database. The RESD-SQL model also failed matching as it commonly employed case sensitive matching, i.e., it failed the NQL: *'Show me all your horror movies'* purely because it tried to match the lower case 'horror' when, in the database, it is uppercase ('HORROR'). DIN-SQL did not fail these matches as in the self-correction stage it is explicitly requested to match using the lower-case forms (using the LOWER operator when required.)

- **Schema Matching** – These occur when the model identifies the wrong database schema items. For example, the RESD-SQL model mismatched the movie type and award fields, or when foreign keys were mismatched.
- **Construction** – These errors occurred when the NLQ required the construction of intermediate values based on calculations, such as when calculating the length of membership for a customer.
- **Incomplete** – This error occurred once in response to the NLQ: *‘Which movie, by ‘Hitchcock, Alfred’ won the most awards?’* Four movies met this criterion however the DIN-SQL model incorrectly limited the results to a single datapoint.
- **Semantic errors** – Semantic errors occurred when the meaning of the query has been lost, usually when an important sub-query is missed.
- **Compound** – Arise from numerous errors in the generated SQL code.

Table 2 below displays summarises the results of the RESDSQL and DIN-SQL approaches to answering the 27 NLQ for the Movies database.

Table 2: Summary of functional testing of the DIN-SQL and RESD-SQL approaches on the Movie database.

	DIN-SQL	RESD-SQL
<u>Matches</u>		
Basic Queries (9 total)	9	5
Advanced Queries (6 total)	6	2
Complex Queries (12 total)	8	3
<u>Errors</u>		
Equality Errors	1	2
Incomplete	1	
Schema Matching		7
Filter Error		2
Semantic Error	2	3
Construction		3

Discussion

This section discusses the results, strength and weaknesses of the two NLIDB approaches.

DIN-SQL Approach

The DIN-SQL inference approach has the following advantages:

- It is lightweight. The total code can be captured in a single script with a few, simple functions.
- It demonstrates high performance.
- It is easy to maintain. The code required to generate the prompts is simple and minimal and is therefore easily modified. The authors easily implemented a few simple changes as documented earlier.
- It doesn't require any significant computing resources.
- Model is fast, completing queries in a few seconds.
- It doesn't require significant pre-processing of the database schema. It performed well using database field names that were only rough abbreviations of their semantic names.

The DIN-SQL inference approach has the following disadvantages:

- It depends on an external service provider (OpenAI). Data regarding the structure of the database (although, not the underlying data) is sent externally, which may be a security concern.
- It may prove expensive at scale (Kaddour, 2023). Being dependent on an external service provider. It requires an API key, with charges per input and output tokens. Cost is usage based, although, the cost may be modest depending on the organisational needs. For example, the functional testing

outlined in the results section required approximately 20,000 tokens in prompts and 1,000 tokens in outputs for a total cost of approximately \$1.30 (United States Dollars).

RESDSQL approach

The RESDSQL approach has the following advantages:

- It can be entirely locally deployed. No database information leaves the host organisation's network.
- It's cost effective, in that it doesn't depend on external services, and the cost shouldn't scale with the number or complexity of queries (cost is capacity based, rather than usage).
- It can be further fine-tuned, against internally derived data.
- It has high accuracy and generalisability (published performance accuracy in Table 1 previously).

The RESDSQL approach has the following disadvantages:

- It is a large, complex program. Significant pre-processing and post-processing are required to get the system to work, with multiple classes and scripts required to get working (Kaddour, 2023).
- It is computationally expensive. It took more than 30 seconds to run a simple query. The authors attempted to undertake fine-tuning; however, the test computer lacked the compute power (Li et. al. used a server with a NVIDIA A100 80G GPU and Intel® Xeon® Silver 4316 CPU, 256 GB).

- The system is difficult to maintain. The code has numerous dependencies, and whenever key libraries (such as pytorch) are updated, the models may need to be re-trained.
- Database schema required meaningful names, prompting the authors to produce the CleanMovies database.

Comparative Discussion

- DIN-SQL performed exceptionally. It displayed very advanced SQL skills, and an ability to break down complex, nested queries, and successfully solve them (including converting dates to Australian format!). It also provided capability that mimics reasoning, being able to successfully interpret for example, that a director is alive if they don't have a date of death, and to be able to successfully interpret the concept of '*tenure*'.
- The RESD-SQL was not as powerful as the DIN-SQL, and performed significantly worse than would be expected given its high performance on the Spider dataset. However, this may be an unfair comparison as the size of the LLMs being employed (GPT-4 vs T5-base) are vastly different and reflect the capability that can be gained by employed cloud-based services versus local deployment.
- Both models had issues when queries required exact matching, this is particularly true when trying to match names, which in the Movies database, had the format 'Last name, first name'.

Conclusion

Recommendation

In conclusion, the DIN-SQL approach is superior compared to the RESDSQL approach and we recommend RMIT University to adopt DIN-SQL for their project. However, there are some things that need to be kept in mind. Firstly, our recommended approach does cost money and if scaled up to thousands of queries per day each year would become quite expensive. Secondly, an IT team is still necessary as the approach only considers the querying of a database. Creation of new tables in the database, deleting obsolete tables and being consistent with data entry (column names, primary keys) still needs to be performed by the IT team. Nevertheless, this project would help to reduce unnecessary requests to the IT department and ensure that less time is wasted on non-essential tasks if implemented (Elondou, 2023). Lastly, there is a potential security constraint in that for the DIN-SQL approach it requires database schema to be transmitted outside of RMIT internal networks. While the underlying data is not transferred, revealing of the structure and contents of the database represents a security vulnerability.

Further study

This work was able to quickly stand up to a NLIDB capability, leveraging cloud based LLMs that are fit for purpose in a RMIT business context. However, the solution could be further enhanced, with the following points as a potential starter for further research and development:

- The DIN-SQL tool revealed the power of using prompting templates with a LLM and shows how specific prompt wording can direct the LLM in useful ways (deep learning).

- Considering the above point, training material should be developed to support users of the tool, guiding them in how to best word their NLQ (prompt engineering).
- A conversational style interface could be implemented that interacts with the GPT-4 model more dynamically (chatbot). Skilled business professionals, who can break down their questions into further sub-questions would then be able to interact with databases more effectively.
- The effectiveness of the other LLMs such as Llama2 should be assessed.

Besides that, in the team's conceptual design there is a front-end interface that is web-based for the user to submit their input to the LLM and will display the generated output to the user once the LLM responds. We have a potential interface that can be used for the project (see Appendix A, Figure 3 and Figure 4) but due to lack of time we are unable to find a way to connect that interface to the LLM back-end (DIN-SQL). Should this project be continued in the future this problem requires attention.

References

- Affolter, K., Stockinger, K., & Bernstein, A. (2019). A Comparative Survey of Recent Natural Language Interfaces for Databases. *The VLDB Journal*, 28(5), 793–819. <https://doi.org/10.1007/s00778-019-00567-8>
- Deriu, J., Mlynchik, K., Schläpfer, P., Rodrigo, A., Von Grünigen, D., Kaiser, N., Stockinger, K., Agirre, E., & Cieliebak, M. (2020). A Methodology for Creating Question Answering Corpora Using Inverse Data Annotation. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 897–911. <https://doi.org/10.18653/v1/2020.acl-main.84>
- Eloundou, T., Manning, S., Mishkin, P. and Rock, D. (2023). GPTs are GPTs: An early look at the labor market impact potential of large language models (arXiv:2303.10130). arXiv. <https://arxiv.org/abs/2303.10130>
- Iacob, R. C. A., Brad, F., Apostol, E.-S., Truică, C.-O., Hosu, I. A., & Rebedea, T. (2020). Neural Approaches for Natural Language Interfaces to Databases: A Survey. *Proceedings of the 28th International Conference on Computational Linguistics*, 381–395. <https://doi.org/10.18653/v1/2020.coling-main.34>
- Kaddour, J., Harris, J., Mozes, M., Bradley, H., Raileanu, R. and McHardy, R. (2023). Challenges and applications of large language models (arXiv:2307.10169). arXiv. <https://arxiv.org/abs/2307.10169>
- LeCun, Yann (2023), Twitter, accessed 23 August 2023. "A survey of LLMs with a practical guide and evolutionary tree". Archived from the original on 23 Jun 2023
- Li, H., Zhang, J., Li, C., & Chen, H. (2023). RESDSQL: Decoupling Schema Linking and Skeleton Parsing for Text-to-SQL (arXiv:2302.05965). arXiv. <http://arxiv.org/abs/2302.05965>
- M, D. (2023).  *CallGPT*. [online] GitHub. Available at: <https://github.com/dminGod/CallGPT>
- platform.openai.com. (n.d.). *OpenAI API*. [online] Available at: <https://platform.openai.com/>
- Pourreza, M., & Rafiei, D. (2023). DIN-SQL: Decomposed In-Context Learning of Text-to-SQL with Self-Correction (arXiv:2304.11015). arXiv. <http://arxiv.org/abs/2304.11015>
- Sumanasekera, S. (2017). ISYS1055/1057 Database Concepts, Week 7: Tutorial/Lab - SQL Programming. RMIT University.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30.

https://proceedings.neurips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

Yu, T., Zhang, R., Yang, K., Yasunaga, M., Wang, D., Li, Z., Ma, J., Li, I., Yao, Q., Roman, S., Zhang, Z., & Radev, D. (2018). Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, 3911–3921. <https://doi.org/10.18653/v1/D18-1425>

Zhao, W.X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z. and Du, Y. (2023). A survey of large language models (arXiv:2303.18223). arXiv. <https://arxiv.org/abs/2303.18223>

Zhong, V., Xiong, C., & Socher, R. (2017). Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning (arXiv:1709.00103). arXiv. <http://arxiv.org/abs/1709.00103>

Appendix A: Test Design

Web-based Front Interface for Display to the User

```
699
700     var ajaxTime= new Date().getTime();
701
702     // This is the actual call to OpenAI
703     $.ajax({
704         type: "POST",
705         url: openai_url,
706         beforeSend: function (xhr) {
707             xhr.setRequestHeader('Authorization', 'Bearer ' + api_key);
708         },
709         data: JSON.stringify(send_message),
710         success: success_behavior,
711         contentType: 'application/json; charset=utf-8',
712         dataType: "json"
713     }).done(function(data) {
714
715         secs = ((new Date().getTime() - ajaxTime)/1000).toFixed(2);
716
717         session_time_taken_secs += parseFloat(secs);
718         session_calls_done += 1;
719
720         session_avg_time_taken = (session_time_taken_secs / session_calls_done).toFixed(2);
721
722         $("#last_time_taken").html(secs + " sec");
723         $("#avg_time_taken").html(session_avg_time_taken + " sec");
724
725     }).fail(function(data) {
726
727         $("#status_message").css("color", "red").html("Error with previous request. Please check the logs or try again.");
728         console.log(data);
729
730         setTimeout(function() {
731             $("#status_message").css("color", "black").html(".");
732             }, 3000);
733     })
734 }
735 }
```

Figure 3: Web interface code that gets ChatGPT instance using Ajax for asynchronous calls through the backend. OpenAI API key is required for this to work. Each time a response is generated it costs some money so be aware.

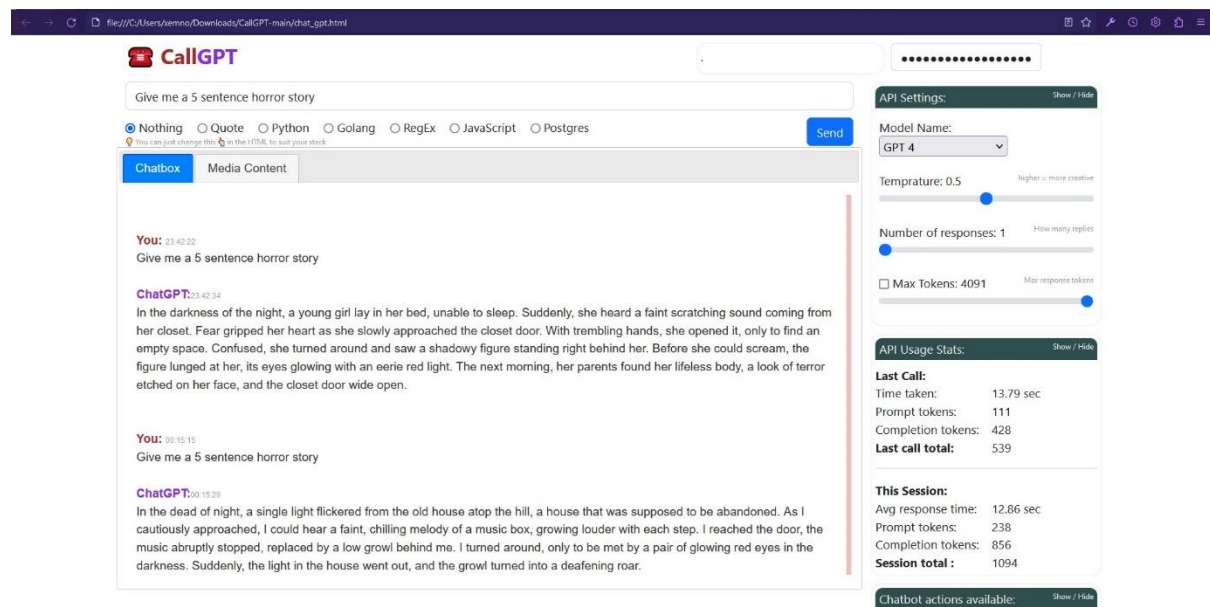


Figure 4: Web interface in action, OpenAI API key is entered in the text field in the top right corner and user input command is entered in the text box to the left below the logo. In this example, a command "Give me a 5-sentence horror story" is given to ChatGPT's GPT 3.5 model for the first response and then to ChatGPT's GPT 4 model for the second response below. As can be seen in the picture, the first response is longer than five sentences while the second response is exactly five sentences. This shows the clear progress and improvements made from an older version of ChatGPT's model to the newer version. (M.D, 2023)

Queries used for Model Evaluation

The following questions were adapted from Tutorials from RMIT's course ISYS1055/1057 Database Concepts (Sumanasekera, 2017), and are based on the database schema shown in Figure 5.

Basic, single table queries:

1. What are the movies available in your collection?
2. Can you show me the movie titles?
3. How many movies are there in your collection.
4. Show me all your horror movies.
5. What horror movies were nominated for academy awards?
6. Display all horror movies with award nominations or comedies.
7. Display all horror and comedy movies that have been nominated for awards.
8. Show me all the movies that have received more than three academy awards.
9. Do you know what movie has the word 'Nuremberg' in it?

Advanced operations queries:

1. On average, how many movies do members rent?
2. Can you prepare a report containing member names and their joining dates?
3. Can you prepare a report containing member names and their joining dates, but display the dates in Australian standard format?
4. Display the members with their join dates and tenures (in both days and years)
5. Display the members with their join dates and tenures (in both days and years), in alphabetical order of names.
6. Display the members with their join dates and tenures (in both days and years), and sort by descending order of tenure.

Complex queries:

1. Display movie titles and director names for all movies
2. Display movies that are directed by Woody Allen
3. Display all movies by directors who are still alive, along with their directors.
4. Who are the stars in the movie "Manhattan"?
5. Display movies whose directors also acted in them.
6. Display a list of directors with the movies they directed.

7. Display a list of actors with the movies they starred in.
8. Show me a list of actors who starred in movies with "Allen, Woody".
9. What is the most recent movie in the collection?
10. What is the oldest movie in the collection?
11. List the movies, which were directed by 'Hitchcock, Alfred' that 'tran, thanh' hasn't borrowed.
12. Which movie, by 'Hitchcock, Alfred' won the most awards?

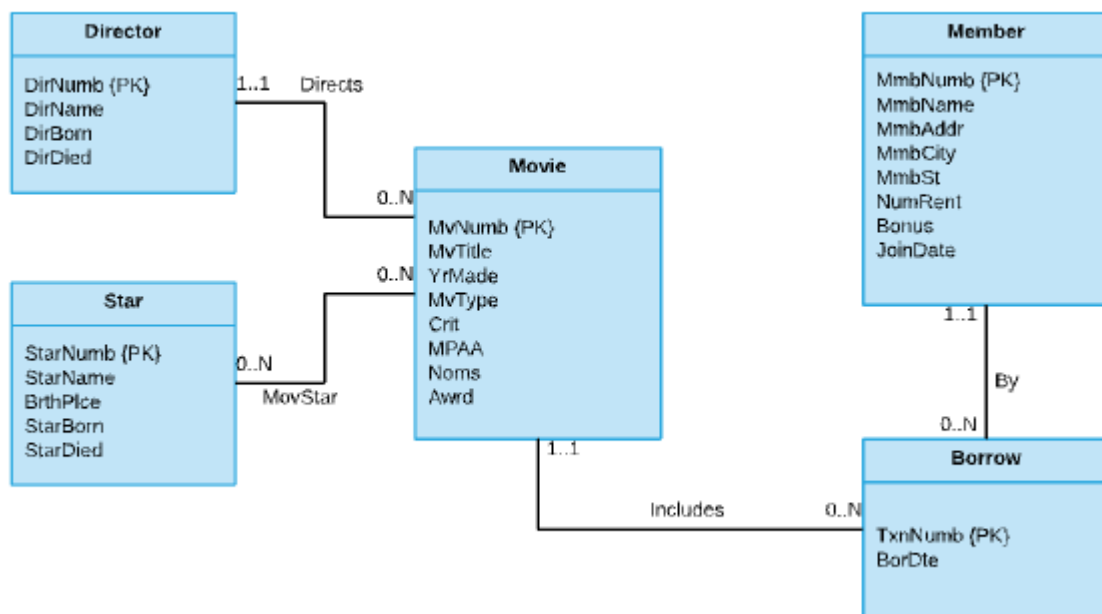


Figure 5: Movies database reproduced from (Sumanasekera, 2017)

Appendix B: Output from NLIDB evaluation scripts

RESD-SQL Evaluation Testing

Basic Queries

Question: 0

Natural language question: What are the movies available in your collection?

Target SQL: SELECT * FROM Movie;

Predicted SQL: select movie.title from MOVIE

Target output:

(1, 'Annie Hall', 1977, 'COMEDY', 4, 'PG', 5, 4, 1)
(2, 'Dr. Strangelove', 1964, 'COMEDY', 4, 'PG', 4, 0, 5)
(3, 'Clockwork Orange', 1971, 'SCI FI', 4, 'R', 3, 0, 5)
(4, 'North by Northwest', 1959, 'SUSPEN', 4, 'PG', 1, 0, 2)
(5, 'Rope', 1948, 'SUSPEN', 3, 'NR', 0, 0, 2)
(6, 'Psycho', 1960, 'HORROR', 4, 'PG', 3, 0, 2)
(7, 'Interiors', 1978, 'DRAMA', 3, 'PG', 3, 0, 1)
(8, 'The Birds', 1963, 'HORROR', 3, 'NR', 0, 0, 2)
(9, 'Samson and Delilah', 1949, 'RELIGI', 2, 'NR', 1, 0, 3)
(10, 'Guess Who's Coming to Dinner', 1967, 'COMEDY', 3, 'NR', 6, 2, 4)
(11, 'Manhattan', 1979, 'COMEDY', 4, 'R', 2, 0, 1)
(12, 'Vertigo', 1958, 'SUSPEN', 4, 'NR', 0, 0, 2)
(13, 'Judgment at Nuremberg', 1961, 'DRAMA', 3, 'NR', 6, 2, 4)
(14, '2001', 1968, 'SCI FI', 4, 'G', 2, 0, 5)
(15, 'The Man with the Golden Arm', 1955, 'DRAMA', 3, 'NR', 1, 0, 6)
(16, 'Anatomy of Murder', 1959, 'SUSPEN', 4, 'NR', 4, 0, 6)
(17, 'Inherit the Wind', 1960, 'DRAMA', 4, 'NR', 2, 0, 4)
(18, 'Laura', 1944, 'SUSPEN', 4, 'NR', 3, 1, 6)
(19, 'The Ten Commandments', 1956, 'RELIGI', 3, 'NR', 1, 0, 3)
(20, 'The Moon is Blue', 1953, 'COMEDY', 2, 'NR', 1, 0, 6)
(21, 'Stagecoach', 1939, 'WESTER', 4, 'NR', 3, 1, 7)
(22, 'Rear Window', 1954, 'SUSPEN', 4, 'NR', 1, 0, 2)
(23, 'Mogambo', 1953, 'WESTER', 3, 'NR', 2, 0, 7)
(24, 'Grapes of Wrath', 1940, 'DRAMA', 4, 'NR', 4, 2, 7)

RESD-SQL output:

('Annie Hall',)	('Judgment at Nuremberg',)
('Dr. Strangelove',)	('2001',)
('Clockwork Orange',)	('The Man with the Golden Arm',)
('North by Northwest',)	('Anatomy of Murder',)
('Rope',)	('Inherit the Wind',)
('Psycho',)	('Laura',)
('Interiors',)	('The Ten Commandments',)
('The Birds',)	('The Moon is Blue',)
('Samson and Delilah',)	('Stagecoach',)
('Guess Who's Coming to Dinner',)	('Rear Window',)
('Manhattan',)	('Mogambo',)
('Vertigo',)	('Grapes of Wrath',)

Question: 1

Natural language question: Can you show me the movie titles?

Target SQL: SELECT mvtitle FROM Movie;

Predicted SQL: select movie.title from MOVIE

Target output:

('Annie Hall',)
('Dr. Strangelove',)
('Clockwork Orange',)
('North by Northwest',)
('Rope',)
('Psycho',)
('Interiors',)
('The Birds',)
('Samson and Delilah',)
("Guess Who's Coming to Dinner",)
('Manhattan',)
('Vertigo',)

('Judgment at Nuremberg',)
('2001',)
('The Man with the Golden Arm',)
('Anatomy of Murder',)
('Inherit the Wind',)
('Laura',)
('The Ten Commandments',)
('The Moon is Blue',)
('Stagecoach',)
('Rear Window',)
('Mogambo',)
('Grapes of Wrath',)

RES-D-SQL output:

('Annie Hall',)
('Dr. Strangelove',)
('Clockwork Orange',)
('North by Northwest',)
('Rope',)
('Psycho',)
('Interiors',)
('The Birds',)
('Samson and Delilah',)
("Guess Who's Coming to Dinner",)
('Manhattan',)
('Vertigo',)

('Judgment at Nuremberg',)
('2001',)
('The Man with the Golden Arm',)
('Anatomy of Murder',)
('Inherit the Wind',)
('Laura',)
('The Ten Commandments',)
('The Moon is Blue',)
('Stagecoach',)
('Rear Window',)
('Mogambo',)
('Grapes of Wrath',)

Question: 2

Natural language question: How many movies are there in your collection

Target SQL: SELECT COUNT(*) FROM Movie;

Predicted SQL: select count(*) from MOVIE

Target output:

(24,)

RES-D-SQL output:

(24,)

Question: 3

Natural language question: Show me all your horror movies

Target SQL: SELECT mvtitle FROM Movie WHERE LOWER(mvtype) = 'horror'

Predicted SQL: select movie.type from MOVIE where movie.type = "Horror"

Target output:

('Psycho',)
('The Birds',)

RES-D-SQL output:

Question: 4

Natural language question: What horror movies were nominated for academy awards

Target SQL: SELECT mvtitle FROM Movie WHERE LOWER(mvtype) = 'horror' AND noms >0
Predicted SQL: select movie.type from MOVIE where movie.awards = "Horror"

Target output:
(('Psycho',))

RES-D-SQL output:

Question: 5

Natural language question: Display all horror movies with award nominations or, comedies.

Target SQL: SELECT mvtitle FROM Movie WHERE (LOWER(mvtype) = 'horror' AND noms >0) OR LOWER(mvtype) = 'comedy'

Predicted SQL: select movie.type from MOVIE where movie.awards = "Horror" or movie.awards = "Comedies"

Target output:

('Annie Hall',)	("Guess Who's Coming to Dinner",)
('Dr. Strangelove',)	('Manhattan',)
('Psycho',)	('The Moon is Blue',)

RES-D-SQL output:

Question: 6

Natural language question: Display all horror and comedy movies that have been nominated for awards

Target SQL: SELECT mvtitle FROM Movie WHERE (LOWER(mvtype) = 'horror' OR LOWER(mvtype) = 'comedy') AND noms > 0

Predicted SQL: select movie.type from MOVIE where movie.awards = "Horror" intersect select movie.type from MOVIE where movie.awards = "Comedy"

Target output:

('Annie Hall',)	("Guess Who's Coming to Dinner",)
('Dr. Strangelove',)	('Manhattan',)
('Psycho',)	('The Moon is Blue',)

RES-D-SQL output:

Question: 7

Natural language question: Show me all the movies that have received more than three academy awards

Target SQL: SELECT mvtitle, mvtype, awrd FROM movie WHERE awrd >3

Predicted SQL: select movie.title from MOVIE where movie.awards > 3

Target output:

(('Annie Hall', 'COMEDY', 4))

RES-D-SQL output:

(('Annie Hall',))

Question: 8

Natural language question: Do you know what movie has the word 'Nuremberg' in it?

Target SQL: SELECT mvtitle FROM movie WHERE LOWER(mvtitle) LIKE '%nuremberg%'

Predicted SQL: select movie.title from MOVIE where movie.title like "%Nuremberg%"

Target output:

('Judgment at Nuremberg',)

RESD-SQL output:

('Judgment at Nuremberg',)

Advanced Queries

Question: 0

Natural language question: On average, how many movies do members rent?

Target SQL: SELECT AVG(numrent) FROM member;

Predicted SQL: select avg(member.number_of_rentals) from MEMBER

Target output:

(13.0,)

RESD-SQL output:

(13.0,)

Question: 1

Natural language question: Can you prepare a report containing member names and their joining dates?

Target SQL: SELECT mmbname, joindate FROM member;

Predicted SQL: select member.name , member.date_joined from MEMBER

Target output:

('Allen, Donna', '1991-05-25')	('Shippers, John', '1991-09-02')
('Peterson, Mark', '1990-02-20')	('Franklin, Trudy', '1990-12-13')
('Sanchez, Miguel', '1991-06-21')	('Stein, Shelly', '1991-06-21')
('Tran, Thanh', '1991-07-03')	('Stein, Billy', '1992-02-27')
('Roberts, Terry', '1990-11-16')	('Stein, Angie', '1991-06-21')
('MacDonald, Greg', '1991-01-29')	('Stein, Willy', '1991-06-21')
('VanderJagt, Neal', '1990-08-11')	

RESD-SQL output:

('Allen, Donna', '1991-05-25')	('Shippers, John', '1991-09-02')
('Peterson, Mark', '1990-02-20')	('Franklin, Trudy', '1990-12-13')
('Sanchez, Miguel', '1991-06-21')	('Stein, Shelly', '1991-06-21')
('Tran, Thanh', '1991-07-03')	('Stein, Billy', '1992-02-27')
('Roberts, Terry', '1990-11-16')	('Stein, Angie', '1991-06-21')
('MacDonald, Greg', '1991-01-29')	('Stein, Willy', '1991-06-21')
('VanderJagt, Neal', '1990-08-11')	

Question: 2

Natural language question: Can you prepare a report containing member names and their joining dates, but display the dates in Australian standard format?

Target SQL: SELECT mmbname, strftime("%d/%m/%Y", joindate) FROM member;

Predicted SQL: select member.name , member.date_joined from MEMBER where member.date_joined = "Australian"

Target output:

('Allen, Donna', '25/05/1991')	('Sanchez, Miguel', '21/06/1991')
('Peterson, Mark', '20/02/1990')	('Tran, Thanh', '03/07/1991')

('Roberts, Terry', '16/11/1990')
 ('MacDonald, Greg', '29/01/1991')
 ('VanderJagt, Neal', '11/08/1990')
 ('Shippers, John', '02/09/1991')
 ('Franklin, Trudy', '13/12/1990')

('Stein, Shelly', '21/06/1991')
 ('Stein, Billy', '27/02/1992')
 ('Stein, Angie', '21/06/1991')
 ('Stein, Willy', '21/06/1991')

RES-D-SQL output:

 Question: 3

Natural language question: Display the members with their join dates and tenures (in both days and years)

Target SQL: SELECT mmbname AS "Member Name", strftime('%d/%m/%Y',joindate) AS "Joined Date", strftime('%d/%m/%Y','now') AS "Today's Date", round(strftime('%J','now') - strftime('%J',joindate)) AS "No. of days since joined", round((strftime('%J','now')- strftime('%J',joindate)) / 365) AS "No of years since joined" FROM member;

Predicted SQL: select member.date_joined , member.date_joined from MEMBER

Target output:

('Allen, Donna', '25/05/1991', '25/10/2023', 11841.0, 32.0)
 ('Peterson, Mark', '20/02/1990', '25/10/2023', 12300.0, 34.0)
 ('Sanchez, Miguel', '21/06/1991', '25/10/2023', 11814.0, 32.0)
 ('Tran, Thanh', '03/07/1991', '25/10/2023', 11802.0, 32.0)
 ('Roberts, Terry', '16/11/1990', '25/10/2023', 12031.0, 33.0)
 ('MacDonald, Greg', '29/01/1991', '25/10/2023', 11957.0, 33.0)
 ('VanderJagt, Neal', '11/08/1990', '25/10/2023', 12128.0, 33.0)
 ('Shippers, John', '02/09/1991', '25/10/2023', 11741.0, 32.0)
 ('Franklin, Trudy', '13/12/1990', '25/10/2023', 12004.0, 33.0)
 ('Stein, Shelly', '21/06/1991', '25/10/2023', 11814.0, 32.0)
 ('Stein, Billy', '27/02/1992', '25/10/2023', 11563.0, 32.0)
 ('Stein, Angie', '21/06/1991', '25/10/2023', 11814.0, 32.0)
 ('Stein, Willy', '21/06/1991', '25/10/2023', 11814.0, 32.0)

RES-D-SQL output:

('1991-05-25', '1991-05-25')	('1991-09-02', '1991-09-02')
('1990-02-20', '1990-02-20')	('1990-12-13', '1990-12-13')
('1991-06-21', '1991-06-21')	('1991-06-21', '1991-06-21')
('1991-07-03', '1991-07-03')	('1992-02-27', '1992-02-27')
('1990-11-16', '1990-11-16')	('1991-06-21', '1991-06-21')
('1991-01-29', '1991-01-29')	('1991-06-21', '1991-06-21')
('1990-08-11', '1990-08-11')	

 Question: 4

Natural language question: Display the members with their join dates and tenures (in both days and years), in alphabetical order of names

Target SQL: SELECT mmbname AS "Member Name", strftime('%d/%m/%Y',joindate) AS "Joined Date", strftime('%d/%m/%Y','now') AS "Today's Date", round(strftime('%J','now') - strftime('%J',joindate)) AS "No. of days since joined", round((strftime('%J','now')- strftime('%J',joindate)) / 365) AS "No of years since joined" FROM member ORDER BY mmbname;

Predicted SQL: select member.date_joined , member.date_joined from MEMBER order by member.name asc

Target output:

('Allen, Donna', '25/05/1991', '25/10/2023', 11841.0, 32.0)

('Franklin, Trudy', '13/12/1990', '25/10/2023', 12004.0, 33.0)
 ('MacDonald, Greg', '29/01/1991', '25/10/2023', 11957.0, 33.0)
 ('Peterson, Mark', '20/02/1990', '25/10/2023', 12300.0, 34.0)
 ('Roberts, Terry', '16/11/1990', '25/10/2023', 12031.0, 33.0)
 ('Sanchez, Miguel', '21/06/1991', '25/10/2023', 11814.0, 32.0)
 ('Shippers, John', '02/09/1991', '25/10/2023', 11741.0, 32.0)
 ('Stein, Angie', '21/06/1991', '25/10/2023', 11814.0, 32.0)
 ('Stein, Billy', '27/02/1992', '25/10/2023', 11563.0, 32.0)
 ('Stein, Shelly', '21/06/1991', '25/10/2023', 11814.0, 32.0)
 ('Stein, Willy', '21/06/1991', '25/10/2023', 11814.0, 32.0)
 ('Tran, Thanh', '03/07/1991', '25/10/2023', 11802.0, 32.0)
 ('VanderJagt, Neal', '11/08/1990', '25/10/2023', 12128.0, 33.0)

RES-D-SQL output:

('1991-05-25', '1991-05-25')	('1991-06-21', '1991-06-21')
('1990-12-13', '1990-12-13')	('1992-02-27', '1992-02-27')
('1991-01-29', '1991-01-29')	('1991-06-21', '1991-06-21')
('1990-02-20', '1990-02-20')	('1991-06-21', '1991-06-21')
('1990-11-16', '1990-11-16')	('1991-07-03', '1991-07-03')
('1991-06-21', '1991-06-21')	('1990-08-11', '1990-08-11')
('1991-09-02', '1991-09-02')	

Question: 5

Natural language question: Display the members with their join dates and tenures (in both days and years), and sort by descending order of tenure

Target SQL: SELECT mmbname AS "Member Name", strftime('%d/%m/%Y',joindate) AS "Joined Date", strftime('%d/%m/%Y',now) AS "Today's Date", round(strftime('%J',now) - strftime('%J',joindate)) AS "No. of days since joined", round((strftime('%J',now)- strftime('%J',joindate)) / 365) AS "No of years since joined" FROM member ORDER BY strftime('%J',joindate);

Predicted SQL: select member.date_joined , member.date_joined , * from MEMBER order by member.date_joined desc

Target output:

('Peterson, Mark', '20/02/1990', '25/10/2023', 12300.0, 34.0)
 ('VanderJagt, Neal', '11/08/1990', '25/10/2023', 12128.0, 33.0)
 ('Roberts, Terry', '16/11/1990', '25/10/2023', 12031.0, 33.0)
 ('Franklin, Trudy', '13/12/1990', '25/10/2023', 12004.0, 33.0)
 ('MacDonald, Greg', '29/01/1991', '25/10/2023', 11957.0, 33.0)
 ('Allen, Donna', '25/05/1991', '25/10/2023', 11841.0, 32.0)
 ('Sanchez, Miguel', '21/06/1991', '25/10/2023', 11814.0, 32.0)
 ('Stein, Shelly', '21/06/1991', '25/10/2023', 11814.0, 32.0)
 ('Stein, Angie', '21/06/1991', '25/10/2023', 11814.0, 32.0)
 ('Stein, Willy', '21/06/1991', '25/10/2023', 11814.0, 32.0)
 ('Tran, Thanh', '03/07/1991', '25/10/2023', 11802.0, 32.0)
 ('Shippers, John', '02/09/1991', '25/10/2023', 11741.0, 32.0)
 ('Stein, Billy', '27/02/1992', '25/10/2023', 11563.0, 32.0)

RES-D-SQL output:

('1992-02-27', '1992-02-27', 11, 'Stein, Billy', '60 Oakland', 'Cedar', 'In', 2, 0, '1992-02-27')
 ('1991-09-02', '1991-09-02', 8, 'Shippers, John', '208 Grayton', 'Cedar', 'In', 6, 1, '1991-09-02')
 ('1991-07-03', '1991-07-03', 4, 'Tran, Thanh', '108 College', 'Carson', 'In', 3, 0, '1991-07-03')
 ('1991-06-21', '1991-06-21', 3, 'Sanchez, Miguel', '47 Chipwood', 'Hudson', 'Mi', 22, 0, '1991-06-21')
 ('1991-06-21', '1991-06-21', 10, 'Stein, Shelly', '82 Harcourt', 'Hudson', 'Mi', 4, 0, '1991-06-21')
 ('1991-06-21', '1991-06-21', 12, 'Stein, Angie', '82 North', 'Hudson', 'Mi', 14, 0, '1991-06-21')
 ('1991-06-21', '1991-06-21', 13, 'Stein, Willy', '80 Harcourt', 'Hudson', 'Mi', 44, 1, '1991-06-21')

('1991-05-25', '1991-05-25', 1, 'Allen, Donna', '21 Wilson', 'Carson', 'In', 2, 0, '1991-05-25')
 ('1991-01-29', '1991-01-29', 6, 'MacDonald, Greg', '19 Oak', 'Carson', 'In', 11, 1, '1991-01-29')
 ('1990-12-13', '1990-12-13', 9, 'Franklin, Trudy', '103 Bedford', 'Brook', 'Mi', 27, 3, '1990-12-13')
 ('1990-11-16', '1990-11-16', 5, 'Roberts, Terry', '602 Bridge', 'Hudson', 'Mi', 1, 0, '1990-11-16')
 ('1990-08-11', '1990-08-11', 7, 'VanderJagt, Neal', '12 Bishop', 'Mantin', 'Il', 19, 2, '1990-08-11')
 ('1990-02-20', '1990-02-20', 2, 'Peterson, Mark', '215 Raymond', 'Cedar', 'In', 14, 1, '1990-02-20')

Complex Queries

Question: 0

Natural language question: Display movie titles and director names for all movies

Target SQL: SELECT m.mvtitle, d.dirname FROM movie m JOIN director d ON m.dirnumb = d.dirnumb

Predicted SQL: select movie.title , director.name from DIRECTOR join MOVIE on DIRECTOR.director_id = MOVIE.director_id

Target output:

('Annie Hall', 'Allen, Woody')	('Judgment at Nuremberg', 'Kramer, Stanley')
('Dr. Strangelove', 'Kubrick, Stanley')	('2001', 'Kubrick, Stanley')
('Clockwork Orange', 'Kubrick, Stanley')	('The Man with the Golden Arm', 'Preminger, Otto')
('North by Northwest', 'Hitchcock, Alfred')	('Anatomy of Murder', 'Preminger, Otto')
('Rope', 'Hitchcock, Alfred')	('Inherit the Wind', 'Kramer, Stanley')
('Psycho', 'Hitchcock, Alfred')	('Laura', 'Preminger, Otto')
('Interiors', 'Allen, Woody')	('The Ten Commandments', 'De Mille, Cecil B')
('The Birds', 'Hitchcock, Alfred')	('The Moon is Blue', 'Preminger, Otto')
('Samson and Delilah', 'De Mille, Cecil B')	('Stagecoach', 'Ford, John')
('Guess Who's Coming to Dinner', 'Kramer, Stanley')	('Rear Window', 'Hitchcock, Alfred')
('Manhattan', 'Allen, Woody')	('Mogambo', 'Ford, John')
('Vertigo', 'Hitchcock, Alfred')	('Grapes of Wrath', 'Ford, John')

RESQD-SQL output:

('Annie Hall', 'Allen, Woody')	('Judgment at Nuremberg', 'Kramer, Stanley')
('Dr. Strangelove', 'Kubrick, Stanley')	('2001', 'Kubrick, Stanley')
('Clockwork Orange', 'Kubrick, Stanley')	('The Man with the Golden Arm', 'Preminger, Otto')
('North by Northwest', 'Hitchcock, Alfred')	('Anatomy of Murder', 'Preminger, Otto')
('Rope', 'Hitchcock, Alfred')	('Inherit the Wind', 'Kramer, Stanley')
('Psycho', 'Hitchcock, Alfred')	('Laura', 'Preminger, Otto')
('Interiors', 'Allen, Woody')	('The Ten Commandments', 'De Mille, Cecil B')
('The Birds', 'Hitchcock, Alfred')	('The Moon is Blue', 'Preminger, Otto')
('Samson and Delilah', 'De Mille, Cecil B')	('Stagecoach', 'Ford, John')
('Guess Who's Coming to Dinner', 'Kramer, Stanley')	('Rear Window', 'Hitchcock, Alfred')
('Manhattan', 'Allen, Woody')	('Mogambo', 'Ford, John')
('Vertigo', 'Hitchcock, Alfred')	('Grapes of Wrath', 'Ford, John')

Question: 1

Natural language question: Display movies that are directed by Woody Allen

Target SQL: SELECT m.mvtitle, d.dirname FROM movie m JOIN director d ON m.dirnumb = d.dirnumb WHERE lower(d.dirname) = 'allen, woody' OR lower(d.dirname) = 'woody Allen';

Predicted SQL: select movie.title from DIRECTOR join MOVIE on DIRECTOR.director_id = MOVIE.director_id where director.name = "Woody Allen"

Target output:

('Annie Hall', 'Allen, Woody')
 ('Interiors', 'Allen, Woody')
 ('Manhattan', 'Allen, Woody')

RESD-SQL output:

Question: 2

Natural language question: Display all movies by directors who are still alive, along with their directors
Target SQL: SELECT m.mvtitle, d.dirname FROM movie m JOIN director d ON m.dirnumb = d.dirnumb
WHERE d.dirdied IS NULL;

Predicted SQL: select movie.title , director.name from DIRECTOR join MOVIE on
DIRECTOR.director_id = MOVIE.director_id

Target output:

('Annie Hall', 'Allen, Woody')	('2001', 'Kubrick, Stanley')
('Dr. Strangelove', 'Kubrick, Stanley')	('The Man with the Golden Arm', 'Preminger, Otto')
('Clockwork Orange', 'Kubrick, Stanley')	
('Interiors', 'Allen, Woody')	('Anatomy of Murder', 'Preminger, Otto')
('Guess Who's Coming to Dinner', 'Kramer, Stanley')	('Inherit the Wind', 'Kramer, Stanley')
	('Laura', 'Preminger, Otto')
('Manhattan', 'Allen, Woody')	('The Moon is Blue', 'Preminger, Otto')
('Judgment at Nuremberg', 'Kramer, Stanley')	

RESD-SQL output:

('Annie Hall', 'Allen, Woody')	('Judgment at Nuremberg', 'Kramer, Stanley')
('Dr. Strangelove', 'Kubrick, Stanley')	('2001', 'Kubrick, Stanley')
('Clockwork Orange', 'Kubrick, Stanley')	('The Man with the Golden Arm', 'Preminger, Otto')
('North by Northwest', 'Hitchcock, Alfred')	
('Rope', 'Hitchcock, Alfred')	('Anatomy of Murder', 'Preminger, Otto')
('Psycho', 'Hitchcock, Alfred')	('Inherit the Wind', 'Kramer, Stanley')
('Interiors', 'Allen, Woody')	('Laura', 'Preminger, Otto')
('The Birds', 'Hitchcock, Alfred')	('The Ten Commandments', 'De Mille, Cecil B')
('Samson and Delilah', 'De Mille, Cecil B')	('The Moon is Blue', 'Preminger, Otto')
('Guess Who's Coming to Dinner', 'Kramer, Stanley')	('Stagecoach', 'Ford, John')
('Manhattan', 'Allen, Woody')	('Rear Window', 'Hitchcock, Alfred')
('Vertigo', 'Hitchcock, Alfred')	('Mogambo', 'Ford, John')
	('Grapes of Wrath', 'Ford, John')

Question: 3

Natural language question: Who are the stars in the movie "Manhattan"?

Target SQL: SELECT s.starname FROM (movie m JOIN movstar ms ON m.mvnumb = ms.mvnumb)
JOIN star s ON ms.starnumb = s.starnumb WHERE lower(mvtitle) = 'manhattan';

Predicted SQL: select star.name from STAR join MOVIE on STAR.movie_star_id = MOVIE.movie_id
where movie.title = "Manhattan"

Target output:

('Allen, Woody',)
('Keaton, Diane',)

RESD-SQL output:

('Taylor, Rod',)

Question: 4

Natural language question: Display movies whose directors also acted in them

Target SQL: SELECT m.mvtitle FROM ((movie m JOIN movstar ms ON m.mvnumb = ms.mvnumb)
JOIN star s ON ms.starnumb = s.starnumb) JOIN director d ON m.dirnumb = d.dirnumb WHERE
lower(s.starname) = lower (d.dirname);

Predicted SQL: select movie.title from DIRECTOR join MOVIE on DIRECTOR.director_id =
MOVIE.director_id

Target output:

('Annie Hall',)
('Manhattan',)

RESD-SQL output:

('Annie Hall',)	('Judgment at Nuremberg',)
('Dr. Strangelove',)	('2001',)
('Clockwork Orange',)	('The Man with the Golden Arm',)
('North by Northwest',)	('Anatomy of Murder',)
('Rope',)	('Inherit the Wind',)
('Psycho',)	('Laura',)
('Interiors',)	('The Ten Commandments',)
('The Birds',)	('The Moon is Blue',)
('Samson and Delilah',)	('Stagecoach',)
('Guess Who's Coming to Dinner',)	('Rear Window',)
('Manhattan',)	('Mogambo',)
('Vertigo',)	('Grapes of Wrath',)

Question: 5

Natural language question: Display a list of directors with the movies they directed.

Target SQL: SELECT d.dirname, m.mvtitle FROM director d LEFT OUTER JOIN movie m ON d.dirnumb = m.dirnumb;

Predicted SQL: select director.name , movie.title from DIRECTOR join MOVIE on DIRECTOR.director_id = MOVIE.director_id

Target output:

('Allen, Woody', 'Annie Hall')	('Kramer, Stanley', 'Judgment at Nuremberg')
('Allen, Woody', 'Interiors')	('Kubrick, Stanley', '2001')
('Allen, Woody', 'Manhattan')	('Kubrick, Stanley', 'Clockwork Orange')
('Hitchcock, Alfred', 'North by Northwest')	('Kubrick, Stanley', 'Dr. Strangelove')
('Hitchcock, Alfred', 'Psycho')	('Preminger, Otto', 'Anatomy of Murder')
('Hitchcock, Alfred', 'Rear Window')	('Preminger, Otto', 'Laura')
('Hitchcock, Alfred', 'Rope')	('Preminger, Otto', 'The Man with the Golden Arm')
('Hitchcock, Alfred', 'The Birds')	('Preminger, Otto', 'The Moon is Blue')
('Hitchcock, Alfred', 'Vertigo')	('Ford, John', 'Grapes of Wrath')
('De Mille, Cecil B', 'Samson and Delilah')	('Ford, John', 'Mogambo')
('De Mille, Cecil B', 'The Ten Commandments')	('Ford, John', 'Stagecoach')
('Kramer, Stanley', 'Guess Who's Coming to Dinner')	('Fellini, Federico', 'None')
('Kramer, Stanley', 'Inherit the Wind')	

RESD-SQL output:

('Allen, Woody', 'Annie Hall')	('Kramer, Stanley', 'Judgment at Nuremberg')
('Kubrick, Stanley', 'Dr. Strangelove')	('Kubrick, Stanley', '2001')
('Kubrick, Stanley', 'Clockwork Orange')	('Preminger, Otto', 'The Man with the Golden Arm')
('Hitchcock, Alfred', 'North by Northwest')	('Preminger, Otto', 'Anatomy of Murder')
('Hitchcock, Alfred', 'Rope')	('Kramer, Stanley', 'Inherit the Wind')
('Hitchcock, Alfred', 'Psycho')	('Preminger, Otto', 'Laura')
('Allen, Woody', 'Interiors')	('De Mille, Cecil B', 'The Ten Commandments')
('Hitchcock, Alfred', 'The Birds')	('Preminger, Otto', 'The Moon is Blue')
('De Mille, Cecil B', 'Samson and Delilah')	('Ford, John', 'Stagecoach')
('Kramer, Stanley', 'Guess Who's Coming to Dinner')	('Hitchcock, Alfred', 'Rear Window')
('Allen, Woody', 'Manhattan')	('Ford, John', 'Mogambo')
('Hitchcock, Alfred', 'Vertigo')	('Ford, John', 'Grapes of Wrath')

Question: 6

Natural language question: Display a list of actors with the movies they starred in.

Target SQL: SELECT s.starname, m.mvtitle FROM (star s LEFT OUTER JOIN movstar ms ON s.starnumb = ms.starnumb) LEFT OUTER JOIN movie m ON ms.mvnumb = m.mvnumb;
 Predicted SQL: select star.name , movie.title from STAR join MOVIE on STAR.movie_star_id = MOVIE.movie_id

Target output:

('Allen, Woody', 'Annie Hall')	('Tracy, Spencer', 'Inherit the Wind')
('Allen, Woody', 'Manhattan')	('Hepburn, Katharine', 'Guess Who's Coming to Dinner')
('Keaton, Diane', 'Annie Hall')	('Dullea, Keir', '2001')
('Keaton, Diane', 'Interiors')	('Novak, Kim', 'Vertigo')
('Keaton, Diane', 'Manhattan')	('Novak, Kim', 'The Man with the Golden Arm')
('Sellers, Peter', 'Dr. Strangelove')	('Sinatra, Frank', 'The Man with the Golden Arm')
('Scott, George C.', 'Dr. Strangelove')	('March, Frederick', 'Inherit the Wind')
('McDowell, Malcolm', 'Clockwork Orange')	('Andrews, Dana', 'Laura')
('Grant, Cary', 'North by Northwest')	('Heston, Charlton', 'The Ten Commandments')
('Saint, Eva Marie', 'North by Northwest')	('McNamara, Maggie', 'The Moon is Blue')
('Stewart, James', 'Rope')	('Niven, David', 'The Moon is Blue')
('Stewart, James', 'Vertigo')	('Wayne, John', 'Stagecoach')
('Stewart, James', 'Anatomy of Murder')	('Gable, Clark', 'Mogambo')
('Stewart, James', 'Rear Window')	('Kelly, Grace', 'Rear Window')
('Perkins, Anthony', 'Psycho')	('Kelly, Grace', 'Mogambo')
('Leigh, Janet', 'Psycho')	('Fonda, Henry', 'Grapes of Wrath')
('Taylor, Rod', 'The Birds')	('Chaplin, Charlie', None)
('Hedren, Tippi', 'The Birds')	('Keaton, Buster', None)
('Mature, Victor', 'Samson and Delilah')	('Cannet, Geoffrey', None)
('Tracy, Spencer', 'Guess Who's Coming to Dinner')	
('Tracy, Spencer', 'Judgment at Nuremberg')	

RESQD-SQL output:

('Allen, Woody', 'Annie Hall')	('Mature, Victor', 'Judgment at Nuremberg')
('Keaton, Diane', 'Dr. Strangelove')	('Tracy, Spencer', '2001')
('Sellers, Peter', 'Clockwork Orange')	('Hepburn, Katharine', 'The Man with the Golden Arm')
('Scott, George C.', 'North by Northwest')	('Dullea, Keir', 'Anatomy of Murder')
('McDowell, Malcolm', 'Rope')	('Novak, Kim', 'Inherit the Wind')
('Grant, Cary', 'Psycho')	('Sinatra, Frank', 'Laura')
('Saint, Eva Marie', 'Interiors')	('March, Frederick', 'The Ten Commandments')
('Stewart, James', 'The Birds')	('Andrews, Dana', 'The Moon is Blue')
('Perkins, Anthony', 'Samson and Delilah')	('Heston, Charlton', 'Stagecoach')
('Leigh, Janet', 'Guess Who's Coming to Dinner')	('McNamara, Maggie', 'Rear Window')
('Taylor, Rod', 'Manhattan')	('Niven, David', 'Mogambo')
('Hedren, Tippi', 'Vertigo')	('Wayne, John', 'Grapes of Wrath')

 Question: 7

Natural language question: Show me a list of actors who starred in movies with "Allen, Woody"

Target SQL: SELECT s2.starname FROM ((star s1 JOIN movstar ms1 ON s1.starnumb = ms1.starnumb) JOIN movstar ms2 ON ms1.mvnumb = ms2.mvnumb) JOIN star s2 ON ms2.starnumb = s2.starnumb WHERE lower(s1.starname) = 'allen, woody';

Predicted SQL: select star.name from STAR join MOVIE on STAR.movie_star_id = MOVIE.movie_id where movie.title = "Allen , Woody"

Target output:

('Allen, Woody',)
 ('Keaton, Diane',)
 ('Allen, Woody',)
 ('Keaton, Diane',)

RESD-SQL output:

Question: 8

Natural language question: What is the most recent movie in the collection?

Target SQL: SELECT mvtitle FROM movie WHERE yrmde = (SELECT MAX(yrmde) FROM movie);

Predicted SQL: select movie.title from MOVIE order by movie.year_made desc limit 1

Target output:
(('Manhattan',))

RESD-SQL output:
(('Manhattan',))

Question: 9

Natural language question: What is the oldest movie in the collection?

Target SQL: SELECT mvtitle FROM movie WHERE yrmde = (SELECT MIN(yrmde) FROM movie);

Predicted SQL: select movie.title from MOVIE order by movie.year_made desc limit 1

Target output:
(('Stagecoach',))

RESD-SQL output:
(('Manhattan',))

Question: 10

Natural language question: List the movies, which were directed by 'Hitchcock, Alfred' that 'tran, thanh' hasn't borrowed

Target SQL: SELECT mvtitle FROM director d join movie m on d.dirnumb = m.DIRNUMB WHERE d.dirname = 'Hitchcock, Alfred' AND m.mvnumb NOT IN (SELECT mvnumb FROM borrow b JOIN member mm ON b.mmbnumb = mm.mmbnumb WHERE lower(mm.mmbname) = 'tran, thanh')

Predicted SQL: select movie.title from borrow join director join movie where director.name = "Hitchcock , Alfred" and borrow.date_borrowed = "null"

Target output:
(('North by Northwest',))
(('Psycho',))
(('The Birds',))
(('Rear Window',))

RESD-SQL output:

Question: 11

Natural language question: Which movie, by 'Hitchcock, Alfred' won the most awards?

Target SQL: SELECT m.mvtitle, m.awrd FROM movie m JOIN director d ON m.dirnumb = d.dirnumb WHERE m.awrd = (SELECT MAX(awrd) FROM director di JOIN movie mi ON di.dirnumb = mi.dirnumb WHERE di.dirname = 'Hitchcock, Alfred') AND lower(d.dirname) = 'hitchcock, alfred';

Predicted SQL: select movie.title from MOVIE where movie.awards = "Hitchcock , Alfred" group by movie.awards order by count(*) desc limit 1

Target output:

(('North by Northwest', 0))
(('Rope', 0))
(('Psycho', 0))
(('The Birds', 0))

(('Vertigo', 0))
(('Rear Window', 0))

RESD-SQL output:

Basic Queries

Question: 0

Natural language question: What are the movies available in your collection?

Target SQL: SELECT * FROM Movie;

Predicted SQL: SELECT MVTITLE FROM MOVIE

Target output:

(1, 'Annie Hall', 1977, 'COMEDY', 4, 'PG', 5, 4, 1)
(2, 'Dr. Strangelove', 1964, 'COMEDY', 4, 'PG', 4, 0, 5)
(3, 'Clockwork Orange', 1971, 'SCI FI', 4, 'R', 3, 0, 5)
(4, 'North by Northwest', 1959, 'SUSPEN', 4, 'PG', 1, 0, 2)
(5, 'Rope', 1948, 'SUSPEN', 3, 'NR', 0, 0, 2)
(6, 'Psycho', 1960, 'HORROR', 4, 'PG', 3, 0, 2)
(7, 'Interiors', 1978, 'DRAMA', 3, 'PG', 3, 0, 1)
(8, 'The Birds', 1963, 'HORROR', 3, 'NR', 0, 0, 2)
(9, 'Samson and Delilah', 1949, 'RELIGI', 2, 'NR', 1, 0, 3)
(10, 'Guess Who's Coming to Dinner', 1967, 'COMEDY', 3, 'NR', 6, 2, 4)
(11, 'Manhattan', 1979, 'COMEDY', 4, 'R', 2, 0, 1)
(12, 'Vertigo', 1958, 'SUSPEN', 4, 'NR', 0, 0, 2)
(13, 'Judgment at Nuremberg', 1961, 'DRAMA', 3, 'NR', 6, 2, 4)
(14, '2001', 1968, 'SCI FI', 4, 'G', 2, 0, 5)
(15, 'The Man with the Golden Arm', 1955, 'DRAMA', 3, 'NR', 1, 0, 6)
(16, 'Anatomy of Murder', 1959, 'SUSPEN', 4, 'NR', 4, 0, 6)
(17, 'Inherit the Wind', 1960, 'DRAMA', 4, 'NR', 2, 0, 4)
(18, 'Laura', 1944, 'SUSPEN', 4, 'NR', 3, 1, 6)
(19, 'The Ten Commandments', 1956, 'RELIGI', 3, 'NR', 1, 0, 3)
(20, 'The Moon is Blue', 1953, 'COMEDY', 2, 'NR', 1, 0, 6)
(21, 'Stagecoach', 1939, 'WESTER', 4, 'NR', 3, 1, 7)
(22, 'Rear Window', 1954, 'SUSPEN', 4, 'NR', 1, 0, 2)
(23, 'Mogambo', 1953, 'WESTER', 3, 'NR', 2, 0, 7)
(24, 'Grapes of Wrath', 1940, 'DRAMA', 4, 'NR', 4, 2, 7)

GPT output:

('Annie Hall',)	('Judgment at Nuremberg',)
('Dr. Strangelove',)	('2001',)
('Clockwork Orange',)	('The Man with the Golden Arm',)
('North by Northwest',)	('Anatomy of Murder',)
('Rope',)	('Inherit the Wind',)
('Psycho',)	('Laura',)
('Interiors',)	('The Ten Commandments',)
('The Birds',)	('The Moon is Blue',)
('Samson and Delilah',)	('Stagecoach',)
('Guess Who's Coming to Dinner',)	('Rear Window',)
('Manhattan',)	('Mogambo',)
('Vertigo',)	('Grapes of Wrath',)

Question: 1

Natural language question: Can you show me the movie titles?

Target SQL: SELECT mvtitle FROM Movie;

Predicted SQL: SELECT MVTITLE FROM MOVIE

Target output:

('Annie Hall',)
('Dr. Strangelove',)
('Clockwork Orange',)
('North by Northwest',)
('Rope',)
('Psycho',)
('Interiors',)
('The Birds',)
('Samson and Delilah',)
("Guess Who's Coming to Dinner",)
('Manhattan',)
('Vertigo',)

('Judgment at Nuremberg',)
('2001',)
('The Man with the Golden Arm',)
('Anatomy of Murder',)
('Inherit the Wind',)
('Laura',)
('The Ten Commandments',)
('The Moon is Blue',)
('Stagecoach',)
('Rear Window',)
('Mogambo',)
('Grapes of Wrath',)

GPT output:

('Annie Hall',)
('Dr. Strangelove',)
('Clockwork Orange',)
('North by Northwest',)
('Rope',)
('Psycho',)
('Interiors',)
('The Birds',)
('Samson and Delilah',)
("Guess Who's Coming to Dinner",)
('Manhattan',)
('Vertigo',)

('Judgment at Nuremberg',)
('2001',)
('The Man with the Golden Arm',)
('Anatomy of Murder',)
('Inherit the Wind',)
('Laura',)
('The Ten Commandments',)
('The Moon is Blue',)
('Stagecoach',)
('Rear Window',)
('Mogambo',)
('Grapes of Wrath',)

Question: 2

Natural language question: How many movies are there in your collection

Target SQL: SELECT COUNT(*) FROM Movie;

Predicted SQL: SELECT COUNT(DISTINCT MVNUMB) AS 'Number of Movies' FROM MOVIE

Target output:

(24,)

GPT output:

(24,)

Question: 3

Natural language question: Show me all your horror movies

Target SQL: SELECT mvtitle FROM Movie WHERE LOWER(mvtype) = 'horror'

Predicted SQL: SELECT * FROM MOVIE WHERE LOWER(MVTYPE) = 'horror'

Target output:

('Psycho',)
('The Birds',)

GPT output:

(6, 'Psycho', 1960, 'HORROR', 4, 'PG', 3, 0, 2)
(8, 'The Birds', 1963, 'HORROR', 3, 'NR', 0, 0, 2)

Question: 4

Natural language question: What horror movies were nominated for academy awards

Target SQL: SELECT mvtitle FROM Movie WHERE LOWER(mvtype) = 'horror' AND noms > 0

Predicted SQL: SELECT MVTITLE FROM MOVIE WHERE LOWER(MVTYPE) = 'horror' AND NOMS > 0

Target output:

('Psycho',)

GPT output:

('Psycho',)

-----Question: 5
Natural language question: Display all horror movies with award nominations or, comedies.
Target SQL: SELECT mvtitle FROM Movie WHERE (LOWER(mvtype) = 'horror' AND noms > 0) OR LOWER(mvtype) = 'comedy'
Predicted SQL: SELECT MVTITLE FROM MOVIE WHERE (LOWER(MVTYPE) = 'horror' AND NOMS > 0) OR LOWER(MVTYPE) = 'comedy'

Target output:

('Annie Hall',)

('Dr. Strangelove',)('Psycho',)

('Guess Who's Coming to Dinner',)

('Manhattan',)

('The Moon is Blue',)

GPT output:

('Annie Hall',)

('Dr. Strangelove',)('Psycho',)

('Guess Who's Coming to Dinner',)

('Manhattan',)

('The Moon is Blue',)

Question: 6

Natural language question: Display all horror and comedy movies that have been nominated for awards
Target SQL: SELECT mvtitle FROM Movie WHERE (LOWER(mvtype) = 'horror' OR LOWER(mvtype) = 'comedy') AND noms > 0

Predicted SQL: SELECT DISTINCT MVTITLE FROM MOVIE WHERE (LOWER(MVTYPE) = 'horror' OR LOWER(MVTYPE) = 'comedy') AND NOMS IS NOT NULL

Target output:

('Annie Hall',)

('Dr. Strangelove',)

('Psycho',)

('Guess Who's Coming to Dinner',)

('Manhattan',)

('The Moon is Blue',)

GPT output:

('Annie Hall',)

('Dr. Strangelove',)

('Psycho',)

('The Birds',)

('Guess Who's Coming to Dinner',)

('Manhattan',)

('The Moon is Blue',)

Question: 7

Natural language question: Show me all the movies that have received more than three academy awards

Target SQL: SELECT mvtitle, mvtype, awrd FROM movie WHERE awrd > 3

Predicted SQL: SELECT * FROM MOVIE WHERE AWRD > 3

Target output:

('Annie Hall', 'COMEDY', 4)

GPT output:

(1, 'Annie Hall', 1977, 'COMEDY', 4, 'PG', 5, 4, 1)

Question: 8

Natural language question: Do you know what movie has the word 'Nuremberg' in it?
 Target SQL: SELECT mvtitle FROM movie WHERE LOWER(mvtitle) LIKE '%nuremberg%'
 Predicted SQL: SELECT MVTITLE FROM MOVIE WHERE MVTITLE LIKE '%Nuremberg%'

Target output:
 ('Judgment at Nuremberg',)

GPT output:
 ('Judgment at Nuremberg',)

Advanced Queries

Question: 0

Natural language question: On average, how many movies do members rent?

Target SQL: SELECT AVG(numrent) FROM member;

Predicted SQL: SELECT AVG(NUMRENT) FROM MEMBER

Target output:(13.0,)

GPT output:
 (13.0,)

Question: 1

Natural language question: Can you prepare a report containing member names and their joining dates?

Target SQL: SELECT mmbname, joindate FROM member;

Predicted SQL: SELECT MMBNAME, JOINDATE FROM MEMBER

Target output:

('Allen, Donna', '1991-05-25')
 ('Peterson, Mark', '1990-02-20')
 ('Sanchez, Miguel', '1991-06-21')
 ('Tran, Thanh', '1991-07-03')
 ('Roberts, Terry', '1990-11-16')
 ('MacDonald, Greg', '1991-01-29')
 ('VanderJagt, Neal', '1990-08-11')
 ('Shippers, John', '1991-09-02')
 ('Franklin, Trudy', '1990-12-13')
 ('Stein, Shelly', '1991-06-21')
 ('Stein, Billy', '1992-02-27')
 ('Stein, Angie', '1991-06-21')
 ('Stein, Willy', '1991-06-21')

GPT output:

('Allen, Donna', '1991-05-25')
 ('Peterson, Mark', '1990-02-20')
 ('Sanchez, Miguel', '1991-06-21')
 ('Tran, Thanh', '1991-07-03')
 ('Roberts, Terry', '1990-11-16')
 ('MacDonald, Greg', '1991-01-29')
 ('VanderJagt, Neal', '1990-08-11')
 ('Shippers, John', '1991-09-02')
 ('Franklin, Trudy', '1990-12-13')
 ('Stein, Shelly', '1991-06-21')
 ('Stein, Billy', '1992-02-27')
 ('Stein, Angie', '1991-06-21')
 ('Stein, Willy', '1991-06-21')

Question: 2Natural language question: Can you prepare a report containing member names and their joining dates, but display the dates in Australian standard format?

Target SQL: SELECT mmbname, strftime('%d/%m/%Y', joindate) FROM member;

Predicted SQL: SELECT MMBNAME, strftime('%d/%m/%Y', JOINDATE) AS JOINDATE FROM MEMBER

Target output:

('Allen, Donna', '25/05/1991')
 ('Peterson, Mark', '20/02/1990')
 ('Sanchez, Miguel', '21/06/1991')
 ('Tran, Thanh', '03/07/1991')
 ('Roberts, Terry', '16/11/1990')
 ('MacDonald, Greg', '29/01/1991')
 ('VanderJagt, Neal', '11/08/1990')

('Shippers, John', '02/09/1991')
 ('Franklin, Trudy', '13/12/1990')
 ('Stein, Shelly', '21/06/1991')
 ('Stein, Billy', '27/02/1992')
 ('Stein, Angie', '21/06/1991')
 ('Stein, Willy', '21/06/1991')

GPT output:

('Allen, Donna', '25/05/1991')	('Shippers, John', '02/09/1991')
('Peterson, Mark', '20/02/1990')	('Franklin, Trudy', '13/12/1990')
('Sanchez, Miguel', '21/06/1991')	('Stein, Shelly', '21/06/1991')
('Tran, Thanh', '03/07/1991')	('Stein, Billy', '27/02/1992')
('Roberts, Terry', '16/11/1990')	('Stein, Angie', '21/06/1991')
('MacDonald, Greg', '29/01/1991')	('Stein, Willy', '21/06/1991')
('VanderJagt, Neal', '11/08/1990')	

Question: 3

Natural language question: Display the members with their join dates and tenures (in both days and years)

Target SQL: SELECT mmbname AS "Member Name", strftime('%d/%m/%Y', joindate) AS "Joined Date", strftime('%d/%m/%Y', 'now') AS "Today's Date", round(strftime('%J', 'now') - strftime('%J', joindate)) AS "No. of days since joined", round((strftime('%J', 'now') - strftime('%J', joindate)) / 365) AS "No of years since joined" FROM member;

Predicted SQL: SELECT MMBNAME, JOINDATE, julianday('now') - julianday(JOINDATE) AS TENURE_DAYS, (julianday('now') - julianday(JOINDATE))/365 AS TENURE_YEARS FROM MEMBER

Target output:

('Allen, Donna', '25/05/1991', '25/10/2023', 11841.0, 32.0)
('Peterson, Mark', '20/02/1990', '25/10/2023', 12300.0, 34.0)
('Sanchez, Miguel', '21/06/1991', '25/10/2023', 11814.0, 32.0)
('Tran, Thanh', '03/07/1991', '25/10/2023', 11802.0, 32.0)
('Roberts, Terry', '16/11/1990', '25/10/2023', 12031.0, 33.0)
('MacDonald, Greg', '29/01/1991', '25/10/2023', 11957.0, 33.0)
('VanderJagt, Neal', '11/08/1990', '25/10/2023', 12128.0, 33.0)
('Shippers, John', '02/09/1991', '25/10/2023', 11741.0, 32.0)
('Franklin, Trudy', '13/12/1990', '25/10/2023', 12004.0, 33.0)
('Stein, Shelly', '21/06/1991', '25/10/2023', 11814.0, 32.0)
('Stein, Billy', '27/02/1992', '25/10/2023', 11563.0, 32.0)
('Stein, Angie', '21/06/1991', '25/10/2023', 11814.0, 32.0)
('Stein, Willy', '21/06/1991', '25/10/2023', 11814.0, 32.0)

GPT output:

('Allen, Donna', '1991-05-25', 11841.093717071693, 32.44135264951149)
('Peterson, Mark', '1990-02-20', 12300.093717071693, 33.69888689608683)
('Sanchez, Miguel', '1991-06-21', 11814.093717071693, 32.367380046771764)
('Tran, Thanh', '1991-07-03', 11802.093717071693, 32.33450333444299)
('Roberts, Terry', '1990-11-16', 12031.093717071693, 32.961900594716965)
('MacDonald, Greg', '1991-01-29', 11957.093717071693, 32.75916086868957)
('VanderJagt, Neal', '1990-08-11', 12128.093717071693, 33.227654019374505)
('Shippers, John', '1991-09-02', 11741.093717071693, 32.16738004677176)
('Franklin, Trudy', '1990-12-13', 12004.093717071693, 32.88792799197724)
('Stein, Shelly', '1991-06-21', 11814.093717071693, 32.367380046771764)
('Stein, Billy', '1992-02-27', 11563.093717071693, 31.67970881389505)
('Stein, Angie', '1991-06-21', 11814.093717071693, 32.367380046771764)
('Stein, Willy', '1991-06-21', 11814.093717071693, 32.367380046771764)

Question: 4

Natural language question: Display the members with their join dates and tenures (in both days and years), in alphabetical order of names

Target SQL: SELECT mmbname AS "Member Name", strftime('%d/%m/%Y', joindate) AS "Joined Date", strftime('%d/%m/%Y', 'now') AS "Today's Date", round(strftime('%J', 'now') - strftime('%J', joindate)) AS "No. of days since joined", round((strftime('%J', 'now') - strftime('%J', joindate)) / 365) AS "No of years since joined" FROM member ORDER BY mmbname; Predicted SQL: SELECT MMBNAME, JOINDATE,

```
julianday('now') - julianday(JOINDATE) AS TENURE_DAYS, (julianday('now') - julianday(JOINDATE))/365 AS TENURE_YEARS FROM MEMBER ORDER BY MMBNAME ASC
```

Target output:

```
('Allen, Donna', '25/05/1991', '25/10/2023', 11841.0, 32.0)
('Franklin, Trudy', '13/12/1990', '25/10/2023', 12004.0, 33.0)
('MacDonald, Greg', '29/01/1991', '25/10/2023', 11957.0, 33.0)
('Peterson, Mark', '20/02/1990', '25/10/2023', 12300.0, 34.0)
('Roberts, Terry', '16/11/1990', '25/10/2023', 12031.0, 33.0)
('Sanchez, Miguel', '21/06/1991', '25/10/2023', 11814.0, 32.0)
('Shippers, John', '02/09/1991', '25/10/2023', 11741.0, 32.0)
('Stein, Angie', '21/06/1991', '25/10/2023', 11814.0, 32.0)
('Stein, Billy', '27/02/1992', '25/10/2023', 11563.0, 32.0)
('Stein, Shelly', '21/06/1991', '25/10/2023', 11814.0, 32.0)
('Stein, Willy', '21/06/1991', '25/10/2023', 11814.0, 32.0)
('Tran, Thanh', '03/07/1991', '25/10/2023', 11802.0, 32.0)
('VanderJagt, Neal', '11/08/1990', '25/10/2023', 12128.0, 33.0)
```

GPT output:

```
('Allen, Donna', '1991-05-25', 11841.09394225711, 32.44135326645784)
('Franklin, Trudy', '1990-12-13', 12004.09394225711, 32.88792860892359)
('MacDonald, Greg', '1991-01-29', 11957.09394225711, 32.759161485635914)
('Peterson, Mark', '1990-02-20', 12300.09394225711, 33.69888751303318)
('Roberts, Terry', '1990-11-16', 12031.09394225711, 32.96190121166332)
('Sanchez, Miguel', '1991-06-21', 11814.09394225711, 32.36738066371811)
('Shippers, John', '1991-09-02', 11741.09394225711, 32.16738066371811)
('Stein, Angie', '1991-06-21', 11814.09394225711, 32.36738066371811)
('Stein, Billy', '1992-02-27', 11563.09394225711, 31.6797094308414)
('Stein, Shelly', '1991-06-21', 11814.09394225711, 32.36738066371811)
('Stein, Willy', '1991-06-21', 11814.09394225711, 32.36738066371811)
('Tran, Thanh', '1991-07-03', 11802.09394225711, 32.334503951389344)
('VanderJagt, Neal', '1990-08-11', 12128.09394225711, 33.22765463632085)
```

Question: 5

Natural language question: Display the members with their join dates and tenures (in both days and years), and sort by descending order of tenure

Target SQL: `SELECT mmbname AS "Member Name", strftime('%d/%m/%Y',joindate) AS "Joined Date", strftime('%d/%m/%Y','now') AS "Today's Date", round(strftime('%J','now') - strftime('%J',joindate)) AS "No. of days since joined", round((strftime('%J','now')- strftime('%J',joindate)) / 365) AS "No of years since joined" FROM member ORDER BY strftime('%J',joindate);`
 Predicted SQL: `SELECT MMBNAME, JOINDATE, julianday('now') - julianday(JOINDATE) AS TENURE_DAYS, (julianday('now') - julianday(JOINDATE))/365 AS TENURE_YEARS FROM MEMBER ORDER BY TENURE_DAYS DESC`

Target output:

```
('Peterson, Mark', '20/02/1990', '25/10/2023', 12300.0, 34.0)
('VanderJagt, Neal', '11/08/1990', '25/10/2023', 12128.0, 33.0)
('Roberts, Terry', '16/11/1990', '25/10/2023', 12031.0, 33.0)
('Franklin, Trudy', '13/12/1990', '25/10/2023', 12004.0, 33.0)
('MacDonald, Greg', '29/01/1991', '25/10/2023', 11957.0, 33.0)
('Allen, Donna', '25/05/1991', '25/10/2023', 11841.0, 32.0)
('Sanchez, Miguel', '21/06/1991', '25/10/2023', 11814.0, 32.0)
('Stein, Shelly', '21/06/1991', '25/10/2023', 11814.0, 32.0)
('Stein, Angie', '21/06/1991', '25/10/2023', 11814.0, 32.0)
('Stein, Willy', '21/06/1991', '25/10/2023', 11814.0, 32.0)
('Tran, Thanh', '03/07/1991', '25/10/2023', 11802.0, 32.0)
('Shippers, John', '02/09/1991', '25/10/2023', 11741.0, 32.0)
('Stein, Billy', '27/02/1992', '25/10/2023', 11563.0, 32.0)
```

GPT output:

('Peterson, Mark', '1990-02-20', 12300.094143784605, 33.6988880651633)
('VanderJagt, Neal', '1990-08-11', 12128.094143784605, 33.227655188450974)
('Roberts, Terry', '1990-11-16', 12031.094143784605, 32.961901763793435)
('Franklin, Trudy', '1990-12-13', 12004.094143784605, 32.88792916105371)
('MacDonald, Greg', '1991-01-29', 11957.094143784605, 32.75916203776604)
('Allen, Donna', '1991-05-25', 11841.094143784605, 32.441353818587956)
('Sanchez, Miguel', '1991-06-21', 11814.094143784605, 32.36738121584823)
('Stein, Shelly', '1991-06-21', 11814.094143784605, 32.36738121584823)
('Stein, Angie', '1991-06-21', 11814.094143784605, 32.36738121584823)
('Stein, Willy', '1991-06-21', 11814.094143784605, 32.36738121584823)
('Tran, Thanh', '1991-07-03', 11802.094143784605, 32.33450450351947)
('Shippers, John', '1991-09-02', 11741.094143784605, 32.16738121584823)
('Stein, Billy', '1992-02-27', 11563.094143784605, 31.67970998297152)

Complex Queries

Question: 0

Natural language question: Display movie titles and director names for all movies

Target SQL: SELECT m.mvtitle, d.dirname FROM movie m JOIN director d ON m.dirnumb = d.dirnumb

Predicted SQL: SELECT MOVIE.MVTITLE, DIRECTOR.DIRNAME FROM MOVIE JOIN DIRECTOR ON MOVIE.DIRNUMB = DIRECTOR.DIRNUMB

Target output:

('Annie Hall', 'Allen, Woody')
('Dr. Strangelove', 'Kubrick, Stanley')
('Clockwork Orange', 'Kubrick, Stanley')
('North by Northwest', 'Hitchcock, Alfred')
('Rope', 'Hitchcock, Alfred')
('Psycho', 'Hitchcock, Alfred')
('Interiors', 'Allen, Woody')
('The Birds', 'Hitchcock, Alfred')
('Samson and Delilah', 'De Mille, Cecil B')
('Guess Who's Coming to Dinner', 'Kramer, Stanley')
('Manhattan', 'Allen, Woody')
('Vertigo', 'Hitchcock, Alfred')
('Judgment at Nuremberg', 'Kramer, Stanley')
('2001', 'Kubrick, Stanley')
('The Man with the Golden Arm', 'Preminger, Otto')
('Anatomy of Murder', 'Preminger, Otto')
('Inherit the Wind', 'Kramer, Stanley')
('Laura', 'Preminger, Otto')
('The Ten Commandments', 'De Mille, Cecil B')
('The Moon is Blue', 'Preminger, Otto')
('Stagecoach', 'Ford, John')
('Rear Window', 'Hitchcock, Alfred')
('Mogambo', 'Ford, John')
('Grapes of Wrath', 'Ford, John')

GPT output:

('Annie Hall', 'Allen, Woody')
('Dr. Strangelove', 'Kubrick, Stanley')
('Clockwork Orange', 'Kubrick, Stanley')
('North by Northwest', 'Hitchcock, Alfred')
('Rope', 'Hitchcock, Alfred')
('Psycho', 'Hitchcock, Alfred')
('Interiors', 'Allen, Woody')
('The Birds', 'Hitchcock, Alfred')
('Samson and Delilah', 'De Mille, Cecil B')
('Guess Who's Coming to Dinner', 'Kramer, Stanley')
('Manhattan', 'Allen, Woody')
('Vertigo', 'Hitchcock, Alfred')
('Judgment at Nuremberg', 'Kramer, Stanley')
('2001', 'Kubrick, Stanley')
('The Man with the Golden Arm', 'Preminger, Otto')
('Anatomy of Murder', 'Preminger, Otto')
('Inherit the Wind', 'Kramer, Stanley')
('Laura', 'Preminger, Otto')
('The Ten Commandments', 'De Mille, Cecil B')
('The Moon is Blue', 'Preminger, Otto')
('Stagecoach', 'Ford, John')
('Rear Window', 'Hitchcock, Alfred')
('Mogambo', 'Ford, John')
('Grapes of Wrath', 'Ford, John')

Question: 1

Natural language question: Display movies that are directed by Woody Allen

Target SQL: SELECT m.mvtitle, d.dirname FROM movie m JOIN director d ON m.dirnumb = d.dirnumb WHERE lower(d.dirname) = 'allen, woody' OR lower(d.dirname) = 'woody Allen';

Predicted SQL: SELECT MVTITLE FROM MOVIE WHERE DIRNUMB IN (SELECT DIRNUMB FROM DIRECTOR WHERE LOWER(DIRNAME) = LOWER('Woody Allen'))

Target output:

('Annie Hall', 'Allen, Woody')
('Interiors', 'Allen, Woody')
('Manhattan', 'Allen, Woody')

GPT output:

Question: 2

Natural language question: Display all movies by directors who are still alive, along with their directors
Target SQL: SELECT m.mvtitle, d.dirname FROM movie m JOIN director d ON m.dirnumb = d.dirnumb WHERE d.dirdied IS NULL;

Predicted SQL: SELECT MVTITLE, DIRNAME FROM MOVIE JOIN DIRECTOR ON MOVIE.DIRNUMB = DIRECTOR.DIRNUMB WHERE DIRECTOR.DIRDIED IS NULL

Target output:

('Annie Hall', 'Allen, Woody')
('Dr. Strangelove', 'Kubrick, Stanley')
('Clockwork Orange', 'Kubrick, Stanley')
('Interiors', 'Allen, Woody')
("Guess Who's Coming to Dinner", 'Kramer, Stanley')
('Manhattan', 'Allen, Woody')
('Judgment at Nuremberg', 'Kramer, Stanley')
('2001', 'Kubrick, Stanley')
('The Man with the Golden Arm', 'Preminger, Otto')
('Anatomy of Murder', 'Preminger, Otto')
('Inherit the Wind', 'Kramer, Stanley')
('Laura', 'Preminger, Otto')
('The Moon is Blue', 'Preminger, Otto')

GPT output:

('Annie Hall', 'Allen, Woody')
('Dr. Strangelove', 'Kubrick, Stanley')
('Clockwork Orange', 'Kubrick, Stanley')
('Interiors', 'Allen, Woody')
("Guess Who's Coming to Dinner", 'Kramer, Stanley')
('Manhattan', 'Allen, Woody')
('Judgment at Nuremberg', 'Kramer, Stanley')
('2001', 'Kubrick, Stanley')
('The Man with the Golden Arm', 'Preminger, Otto')
('Anatomy of Murder', 'Preminger, Otto')
('Inherit the Wind', 'Kramer, Stanley')
('Laura', 'Preminger, Otto')
('The Moon is Blue', 'Preminger, Otto')

Question: 3

Natural language question: Who are the stars in the movie "Manhattan"?

Target SQL: SELECT s.starname FROM (movie m JOIN movstar ms ON m.mvnumb = ms.mvnumb) JOIN star s ON ms.starnumb = s.starnumb WHERE lower(mvtitle) = 'manhattan';

Predicted SQL: SELECT STAR.STARNAME FROM STAR JOIN MOVSTAR ON STAR.STARNUMB = MOVSTAR.STARNUMB JOIN MOVIE ON MOVSTAR.MVNUMB = MOVIE.MVNUMB WHERE LOWER(MOVIE.MVTITLE) = LOWER('Manhattan')

Target output:

('Allen, Woody',)
('Keaton, Diane',)

GPT output:

('Allen, Woody',)
('Keaton, Diane',)

Question: 4

Natural language question: Display movies whose directors also acted in them
Target SQL: SELECT m.mvtitle FROM ((movie m JOIN movstar ms ON m.mvnumb = ms.mvnumb) JOIN star s ON ms.starnumb = s.starnumb) JOIN director d ON m.dirnumb = d.dirnumb WHERE lower(s.starname) = lower (d.dirname);

Predicted SQL: SELECT DISTINCT MOVIE.MVTITLE FROM MOVIE JOIN MOVSTAR ON MOVIE.MVNUMB = MOVSTAR.MVNUMB WHERE MOVIE.DIRNUMB = MOVSTAR.STARNUMB

Target output:

('Annie Hall',)
('Manhattan',)

GPT output:

('Annie Hall',)
('Clockwork Orange',)
('Manhattan',)

Question: 5

Natural language question: Display a list of directors with the movies they directed.

Target SQL: SELECT d.dirname, m.mvtitle FROM director d LEFT OUTER JOIN movie m ON d.dirnumb = m.dirnumb;

Predicted SQL: SELECT DIRECTOR.DIRNAME, MOVIE.MVTITLE FROM DIRECTOR JOIN MOVIE ON DIRECTOR.DIRNUMB = MOVIE.DIRNUMB

Target output:

('Allen, Woody', 'Annie Hall')
('Allen, Woody', 'Interiors')
('Allen, Woody', 'Manhattan')
('Hitchcock, Alfred', 'North by Northwest')
('Hitchcock, Alfred', 'Psycho')
('Hitchcock, Alfred', 'Rear Window')
('Hitchcock, Alfred', 'Rope')
('Hitchcock, Alfred', 'The Birds')
('Hitchcock, Alfred', 'Vertigo')
('De Mille, Cecil B', 'Samson and Delilah')
('De Mille, Cecil B', 'The Ten Commandments')
('Kramer, Stanley', 'Guess Who's Coming to Dinner')
('Kramer, Stanley', 'Inherit the Wind')
('Kramer, Stanley', 'Judgment at Nuremberg')
('Kubrick, Stanley', '2001')
('Kubrick, Stanley', 'Clockwork Orange')
('Kubrick, Stanley', 'Dr. Strangelove')
('Preminger, Otto', 'Anatomy of Murder')
('Preminger, Otto', 'Laura')
('Preminger, Otto', 'The Man with the Golden Arm')
('Preminger, Otto', 'The Moon is Blue')
('Ford, John', 'Grapes of Wrath')
('Ford, John', 'Mogambo')
('Ford, John', 'Stagecoach')
('Fellini, Federico', None)

GPT output:

('Allen, Woody', 'Annie Hall')
('Kubrick, Stanley', 'Dr. Strangelove')
('Kubrick, Stanley', 'Clockwork Orange')
('Hitchcock, Alfred', 'North by Northwest')
('Hitchcock, Alfred', 'Rope')
('Hitchcock, Alfred', 'Psycho')
('Allen, Woody', 'Interiors')
('Hitchcock, Alfred', 'The Birds')
('De Mille, Cecil B', 'Samson and Delilah')
('Kramer, Stanley', 'Guess Who's Coming to Dinner')
('Allen, Woody', 'Manhattan')
('Hitchcock, Alfred', 'Vertigo')
('Kramer, Stanley', 'Judgment at Nuremberg')
('Kubrick, Stanley', '2001')
('Preminger, Otto', 'The Man with the Golden Arm')
('Preminger, Otto', 'Anatomy of Murder')
('Kramer, Stanley', 'Inherit the Wind')
('Preminger, Otto', 'Laura')
('De Mille, Cecil B', 'The Ten Commandments')
('Preminger, Otto', 'The Moon is Blue')
('Ford, John', 'Stagecoach')
('Hitchcock, Alfred', 'Rear Window')
('Ford, John', 'Mogambo')
('Ford, John', 'Grapes of Wrath')

Question: 6

Natural language question: Display a list of actors with the movies they starred in.

Target SQL: SELECT s.starname, m.mvtitle FROM (star s LEFT OUTER JOIN movstar ms ON s.starnumb = ms.starnumb) LEFT OUTER JOIN movie m ON ms.mvnumb = m.mvnumb;

Predicted SQL: SELECT STAR.STARNAME, MOVIE.MVTITLE FROM STAR JOIN MOVSTAR ON STAR.STARNUMB = MOVSTAR.STARNUMB JOIN MOVIE ON MOVSTAR.MVNUMB = MOVIE.MVNUMB

Target output:

('Allen, Woody', 'Annie Hall')
('Allen, Woody', 'Manhattan')
('Keaton, Diane', 'Annie Hall')
('Keaton, Diane', 'Interiors')
('Keaton, Diane', 'Manhattan')
('Sellers, Peter', 'Dr. Strangelove')
('Scott, George C.', 'Dr. Strangelove')
('McDowell, Malcolm', 'Clockwork Orange')
('Grant, Cary', 'North by Northwest')
('Saint, Eva Marie', 'North by Northwest')
('Stewart, James', 'Rope')
('Stewart, James', 'Vertigo')
('Stewart, James', 'Anatomy of Murder')
('Stewart, James', 'Rear Window')
('Perkins, Anthony', 'Psycho')
('Leigh, Janet', 'Psycho')
('Taylor, Rod', 'The Birds')
('Hedren, Tippi', 'The Birds')
('Mature, Victor', 'Samson and Delilah')
('Tracy, Spencer', 'Guess Who's Coming to Dinner')
('Tracy, Spencer', 'Judgment at Nuremberg')
('Tracy, Spencer', 'Inherit the Wind')
('Hepburn, Katharine', 'Guess Who's Coming to Dinner')
('Dullea, Keir', '2001')
('Novak, Kim', 'Vertigo')
('Novak, Kim', 'The Man with the Golden Arm')
('Sinatra, Frank', 'The Man with the Golden Arm')
('March, Frederick', 'Inherit the Wind')
('Andrews, Dana', 'Laura')
('Heston, Charlton', 'The Ten Commandments')
('McNamara, Maggie', 'The Moon is Blue')
('Niven, David', 'The Moon is Blue')
('Wayne, John', 'Stagecoach')
('Gable, Clark', 'Mogambo')
('Kelly, Grace', 'Rear Window')
('Kelly, Grace', 'Mogambo')
('Fonda, Henry', 'Grapes of Wrath')
('Chaplin, Charlie', None)
('Keaton, Buster', None)
('Cannet, Geoffrey', None)

GPT output:

('Allen, Woody', 'Annie Hall')
('Keaton, Diane', 'Annie Hall')
('Sellers, Peter', 'Dr. Strangelove')
('Scott, George C.', 'Dr. Strangelove')
('McDowell, Malcolm', 'Clockwork Orange')
('Grant, Cary', 'North by Northwest')
('Saint, Eva Marie', 'North by Northwest')
('Stewart, James', 'Rope')
('Perkins, Anthony', 'Psycho')
('Leigh, Janet', 'Psycho')
('Keaton, Diane', 'Interiors')
('Taylor, Rod', 'The Birds')
('Hedren, Tippi', 'The Birds')
('Mature, Victor', 'Samson and Delilah')
('Tracy, Spencer', 'Guess Who's Coming to Dinner')
('Hepburn, Katharine', 'Guess Who's Coming to Dinner')
('Allen, Woody', 'Manhattan')
('Keaton, Diane', 'Manhattan')
('Stewart, James', 'Vertigo')
('Novak, Kim', 'Vertigo')
('Tracy, Spencer', 'Judgment at Nuremberg')
('Dullea, Keir', '2001')
('Novak, Kim', 'The Man with the Golden Arm')
('Sinatra, Frank', 'The Man with the Golden Arm')
('Stewart, James', 'Anatomy of Murder')
('Tracy, Spencer', 'Inherit the Wind')
('March, Frederick', 'Inherit the Wind')
('Andrews, Dana', 'Laura')
('Heston, Charlton', 'The Ten Commandments')
('McNamara, Maggie', 'The Moon is Blue')
('Niven, David', 'The Moon is Blue')
('Wayne, John', 'Stagecoach')
('Stewart, James', 'Rear Window')
('Kelly, Grace', 'Rear Window')
('Gable, Clark', 'Mogambo')
('Kelly, Grace', 'Mogambo')
('Fonda, Henry', 'Grapes of Wrath')

Question: 7

Natural language question: Show me a list of actors who starred in movies with "Allen, Woody"

Target SQL: SELECT s2.starname FROM ((star s1 JOIN movstar ms1 ON s1.starnumb = ms1.starnumb) JOIN movstar ms2 ON ms1.mvnumb = ms2.mvnumb) JOIN star s2 ON ms2.starnumb = s2.starnumb WHERE lower(s1.starname) = 'allen, woody';

Predicted SQL: SELECT STARNAME FROM STAR WHERE STARNUMB IN (SELECT STARNUMB FROM MOVSTAR WHERE MVNUMB IN (SELECT MVNUMB FROM MOVIE JOIN DIRECTOR ON MOVIE.DIRNUMB = DIRECTOR.DIRNUMB WHERE LOWER(DIRECTOR.DIRNAME) = LOWER('Allen, Woody')))

Target output:

('Allen, Woody',)

('Keaton, Diane',)
('Allen, Woody',)
('Keaton, Diane',)

GPT output:

('Allen, Woody',)
('Keaton, Diane',)

Question: 8

Natural language question: What is the most recent movie in the collection?

Target SQL: SELECT mvtitle FROM movie WHERE yrmde = (SELECT MAX(yrmde) FROM movie);

Predicted SQL: SELECT MVTITLE, MAX(YRMDE) FROM MOVIE

Target output:

('Manhattan',)

GPT output:

('Manhattan', 1979)

Question: 9

Natural language question: What is the oldest movie in the collection?

Target SQL: SELECT mvtitle FROM movie WHERE yrmde = (SELECT MIN(yrmde) FROM movie);

Predicted SQL: SELECT MVTITLE, MIN(YRMDE) AS OldestMovieYear FROM MOVIE

Target output:

('Stagecoach',)

GPT output:

('Stagecoach', 1939)

Question: 10

Natural language question: List the movies, which were directed by 'Hitchcock, Alfred' that 'tran, thanh' hasn't borrowed

Target SQL: SELECT mvtitle FROM director d join movie m on d.dirnumb = m.DIRNUMB WHERE d.dirname = 'Hitchcock, Alfred' AND m.mvnumb NOT IN (SELECT mvnumb FROM borrow b JOIN member mm ON b.mmbnumb = mm.mmbnumb WHERE lower(mm.mmbname) = 'tran, thanh')

Predicted SQL: SELECT MOVIE.MVNUMB, MOVIE.MVTITLE FROM MOVIE JOIN DIRECTOR ON MOVIE.DIRNUMB = DIRECTOR.DIRNUMB WHERE LOWER(DIRECTOR.DIRNAME) = LOWER('Hitchcock, Alfred') AND MOVIE.MVNUMB NOT IN (SELECT BORROW.MVNUMB FROM BORROW JOIN MEMBER ON BORROW.MMBNUMB = MEMBER.MMBNUMB WHERE LOWER(MEMBER.MMBNAME) = LOWER('tran, thanh'))

Target output:

('North by Northwest',)
('Psycho',)
('The Birds',)
('Rear Window',)

GPT output:

(4, 'North by Northwest')
(6, 'Psycho')
(8, 'The Birds')
(22, 'Rear Window')

Question: 11

Natural language question: Which movie, by 'Hitchcock, Alfred' won the most awards?

Target SQL: SELECT m.mvtitle, m.awrd FROM movie m JOIN director d ON m.dirnumb = d.dirnumb
WHERE m.awrd = (SELECT MAX(awrd) FROM director di JOIN movie mi ON di.dirnumb = mi.dirnumb
WHERE di.dirname = 'Hitchcock, Alfred') AND lower(d.dirname) = 'hitchcock, alfred'; Predicted SQL:
SELECT MVTITLE FROM MOVIE JOIN DIRECTOR ON MOVIE.DIRNUMB = DIRECTOR.DIRNUMB
WHERE LOWER(DIRNAME) = LOWER('Hitchcock, Alfred') ORDER BY AWRD DESC LIMIT 1

Target output:

('North by Northwest', 0)
('Rope', 0)
('Psycho', 0)
('The Birds', 0)
('Vertigo', 0)
('Rear Window', 0)

GPT output:

('North by Northwest',)