

NATURAL LANGUAGE DATABASE QUERY AND ANALYTICS AGENT

Prajwal G K

Department of ISE, Don Bosco Institute of Technology, Bengaluru, India.

E-mail: gkprajwal5@gmail.com

Mrs. Bhavya B V

Assistant professor Department of ISE, Don Bosco Institute of Technology, Bengaluru, India.

E-mail: bhavya.bv@dbit.co.in

Veeresh

Department of ISE, Don Bosco Institute of Technology, Bengaluru, India.

E-mail: veereshpatil602@gmail.com

Sambram S

Department of ISE, Don Bosco Institute of Technology, Bengaluru, India.

E-mail: sambramappu@gmail.com

Sai Mohan S M

Department of ISE, Don Bosco Institute of Technology, Bengaluru, India.

E-mail: saimohan0095@gmail.com

ABSTRACT

Colleges record semester results in spreadsheets, and answering even a simple question about them, such as how one section performed compared with another, usually means filtering and calculating by hand or writing SQL. This paper presents the Natural Language Database Query and Analytics Agent, a system that lets non-technical users ask such questions in plain English. The agent supplies the schema of a relational database to a large language model, which writes a SQL query. The query is checked, executed in read-only mode, corrected using the database error message if it fails, and the result is returned as text, a table or a chart. The design is applied to the Information Science and Engineering result sheets of 206 students over three semesters (IV, V and VI). Example questions are run against the prepared database, and their answers are reported as ground truth for future testing, along with data-quality problems found in the source sheets. The measured accuracy of the model-generated queries is left for future work.

Keywords: Natural Language Interface, Text-to-SQL, Large Language Model, LLM Agent, Database Query, Data Analytics, Educational Data, Result Analysis.

1. Introduction

Every semester, engineering departments publish result analyses: section-wise marks, pass and fail counts, subject-wise averages and rank lists. Almost all of this information sits in spreadsheets. A faculty member who wants to compare two sections, or to follow one batch from one semester to the next, has to open several workbooks, filter columns and calculate by hand. For a cohort of more than two hundred students this is slow, and

small mistakes are easy to make without noticing.

A relational database and SQL would answer such questions directly, but most faculty and administrative staff do not write SQL. Large language models (LLMs) offer a way around this, since they can translate an English question into a query when they are given the database schema. Public benchmarks such as Spider [1] and BIRD [2] were created to measure exactly this ability.

A generated query still cannot be trusted blindly. It may refer to a column that does not exist, select the wrong rows, or, if left unrestricted, change the stored data.

This paper presents the Natural Language Database Query and Analytics Agent, built around one concrete case: the semester result data of a college department. The agent passes the schema and the question to an LLM, checks what comes back, runs it safely, and shows the answer together with the SQL behind it so that the user can verify it. Section 2 reviews related work, Section 3 describes the system, Section 4 lists its applications, Section 5 discusses challenges, and Section 7 reports what was obtained when the design was applied to real result data.

2. Large Language Models and Text-to-SQL in Data Access

Research on text-to-SQL has moved quickly since LLMs became available. Pourreza and Rafiei [3] studied how far prompting alone can go and proposed DIN-SQL, which splits the task into schema linking, classification of question difficulty, SQL generation and self-correction. They reported execution accuracies of 85.3% on the Spider test set and 55.9% on the BIRD test set. Gao et al. [4] compared prompt designs systematically, looking at how the question is represented and how examples are selected and organised, and combined the best choices into DAIL-SQL, which they reported at 86.6% execution accuracy on Spider.

Several studies deal with what happens after a first attempt fails. Chen et al. [5] showed that a model can be taught to debug its own program by looking at execution results, and reported gains of 2 to 3% on Spider with this approach. Yao et al. [6] proposed ReAct, in which a model alternates between written reasoning and actions on an external environment, and uses the observations to

decide its next step. A query agent follows the same pattern. It acts by running SQL, observes the database response, and revises the query when needed.

These studies are evaluated on benchmark datasets rather than on an institution's own records. They report accuracy figures and do not provide a tool that a faculty member could use. They also say little about preparing data that begins life as a wide spreadsheet. The present work applies the staged prompting of [3] and the error feedback loop of [5] and [6] to that practical setting.

3. Proposed System

The proposed system is a chat-style analytics tool for semester result data. A user types a question, and the system returns an answer in plain language, a table, or a chart, together with the SQL that was executed. The system has five stages, described below and shown in Fig 1.

a. Data Preparation and Storage:

The source workbooks are wide, with one row per student and several columns per subject, and that layout is awkward to query. They are converted into three long-format tables. A students table holds an anonymous identifier and section. A results table holds one row per student and semester with the recorded pass or fail status, total marks and percentage. A marks table holds one row per student, semester and subject with internal, external and total marks. Student names and university seat numbers are replaced by anonymous identifiers before anything is stored, so the language model never sees personal details.

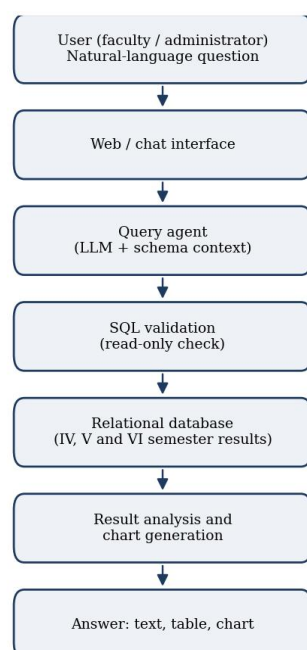


Fig 1: Architecture of the Natural Language Database Query and Analytics Agent

b. Schema-Aware Query Generation:

When a question arrives, the agent builds a prompt containing the table and column descriptions, a few sample rows and a small set of example question and SQL pairs. Following the staged approach of [3], the model is first asked to name the tables and columns it will need and only then to write a single SQL statement. Showing real column names keeps the model from inventing fields.

c. Query Validation and Safe Execution:

Before execution, the statement is checked. Only SELECT statements are accepted, and anything that tries to insert, update, delete or alter data is rejected. Accepted queries run on a read-only database connection, so the stored results cannot be changed through the interface.

d. Error Feedback and Self-Correction:

If the database returns an error, or returns nothing where rows were expected, the question, the failed SQL and the error message are sent back to the model, which is asked for a corrected query. This act, observe and revise cycle follows the ideas

in [5] and [6]. The number of retries is limited so that the user is not kept waiting. The complete workflow is shown in Fig 2.

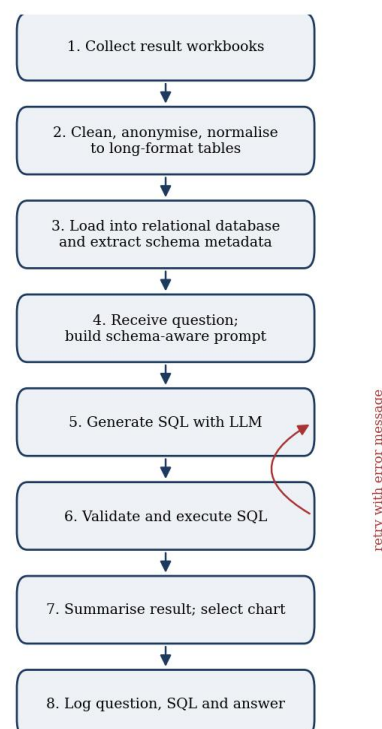


Fig 2: Workflow of the system, including the retry on error

e. Answer and Chart Generation:

The rows returned by the query are passed back to the model, which writes a short answer in plain English. For comparisons and trends, such as pass percentage by section or by semester, the system also chooses a chart type and draws it. The table and the executed SQL are displayed next to the answer, so the user can check how it was obtained.

4. Applications

The system is meant for department staff who know their data but not SQL. It can be used in the following ways.

A. Section-Wise Result Comparison:

Faculty can ask for the pass percentage or the average percentage of any section in any semester, and compare sections without opening each workbook.

B. Subject-Wise Performance Analysis:

Questions such as which subject has the lowest average marks in a semester help course coordinators decide where extra support or revision classes are needed.

C. Cross-Semester Tracking:

Because semesters IV, V and VI are held in one database, the same cohort can be followed over time, for example by comparing average percentages or counting students whose result status changed.

D. Identifying Support Needs:

Counts of students below a chosen threshold, by section and subject, can guide mentoring. Student-level answers should be limited to authorised staff.

E. Preparing Review Material:

Answers, tables and charts produced by the system can be copied into departmental review reports, with the SQL kept as a record of how each figure was obtained.

5. Discussion

LLM-based query tools remove the need to know SQL, but they bring problems of their own. The first is ambiguity. A question such as "who are the best students?" does not say whether best means highest percentage, highest marks in a subject or most improved, and the model has to guess. The second is that terms like pass need a precise definition. In the data used here, the pass or fail status is recorded in each section sheet, and the agent has to use that definition consistently.

A third problem is the quality of the source data, which affects every answer regardless of how good the model is. Section 7 shows that the student sheets and the department's summary sheets do not always agree. Any tool built on spreadsheets inherits such inconsistencies, so it is useful that the agent displays its SQL and the source it used.

There are also limits specific to LLMs. Output can vary between runs, a query that executes without error may still answer a different question from the one asked, and sending data to an external model service raises privacy concerns. The last of these is the reason for anonymising identifiers before storage. Finally, the work is based on one department and three semesters, so results may not carry over to other institutions or data layouts, and the agent still has to be evaluated on a larger set of questions.

6. Conclusion

This paper described a Natural Language Database Query and Analytics Agent for semester result data. It supplies the database schema to an LLM, validates the generated SQL, runs it in read-only mode, corrects itself when errors occur, and presents answers as text, tables or charts together with the executed SQL. The design builds on existing text-to-SQL and agent research and applies it to the practical needs of a college department.

Applied to the result sheets of three semesters, the preparation step produced a database of 612 semester results and 5,508 subject marks, on which a set of example questions was answered. Several data inconsistencies were found in the process. The accuracy of the language model in generating these queries has not yet been measured. Future work will therefore build a larger question set with ground-truth queries, compare different LLMs including smaller locally hosted ones, let the agent ask clarifying questions for ambiguous requests, add role-based access to student-level data, and test the tool with faculty. The system is intended to help staff explore result data, not to replace official result processing, and answers should be checked

against the source records whenever decisions depend on them.

7. Results

This section reports what was obtained by applying the data preparation step and the example queries to the department's result sheets. All SQL in this section was executed on the prepared database, and the answers below are therefore ground-truth values. They are not outputs of the language model.

a. Prepared Database:

The workbooks for semesters IV (2024–25), V (2025–26) and VI contain one sheet per section (A, B and C). After conversion, the database holds the records summarised in Table 1. In total, 206 distinct students appear, of whom 203 are present in all three semesters. Each semester has nine subjects.

Table 1: Records in the prepared database

Semester	Results	Subject marks
IV	205	1,845
V	204	1,836
VI	203	1,827
Total	612	5,508

b. Example Questions and Answers:

Question 1: What is the pass percentage of each section in semester V? The corresponding query is:

```
SELECT section, COUNT(*) AS appeared,
SUM(status='PASS') AS passed,
ROUND(100.0*SUM(status='PASS')/COUNT(*),2)
AS pass_pct
FROM results WHERE semester='V'
GROUP BY section ORDER BY section;
```

It returns 98.59% for section A (70 of 71 students), 94.44% for section B (68 of 72) and 55.74% for section C (34 of 61). Running the same query for each semester gives Table 2 and Fig 3.

Table 2: Pass percentage and average percentage by section

Sem	Sec	Students	Pass %	Avg %
-----	-----	----------	--------	-------

Sem	Sec	Students	Pass %	Avg %
IV	A	70	98.57	85.46
IV	B	72	86.11	80.05
IV	C	63	38.10	65.95
V	A	71	98.59	84.36
V	B	72	94.44	78.93
V	C	61	55.74	68.04
VI	A	70	100.00	87.93
VI	B	72	93.06	82.06
VI	C	61	55.74	69.56

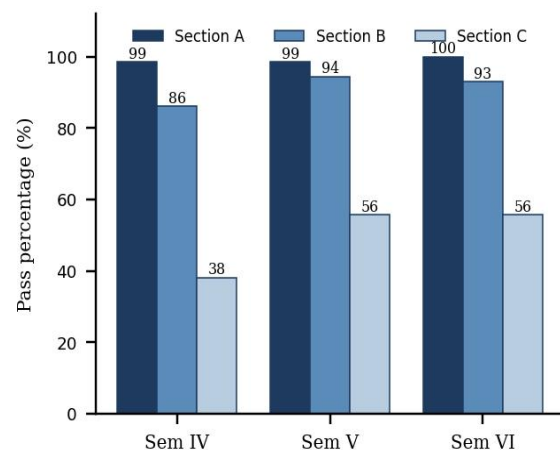


Fig 3: Pass percentage by section and semester, obtained by running Question 1 for each semester

Question 2: Which subject has the lowest average total marks in semester VI? The query averages the total marks of every subject that has an external examination component, orders the result in ascending order and keeps the first row:

```
SELECT subject_code, ROUND(AVG(total),2)
AS avg_total
FROM marks
WHERE semester='VI' AND external IS NOT
NULL
GROUP BY subject_code
ORDER BY avg_total ASC LIMIT 1;
```

The answer is BCS602 with an average of 69.81, followed by BIS601 (72.02) and BIS613D (74.04). The subject with the highest average is the laboratory course BCSL606 (80.81).

Question 3: How many students in each section scored 75% or more in semester V? Using the distinction threshold of 75%

printed on the department's sheets, the answer is 70 in section A, 63 in section B and 5 in section C.

Question 4: How did the average percentage change from semester IV to semester VI for students present in both? A join of the results table with itself on the anonymous identifier covers 203 students. Average percentage rose by 2.48 points in section A, 1.82 in section B and 2.78 in section C. Eight students who passed in semester IV have a failing status in semester VI. The percentages are calculated by the department on different mark totals in different semesters, so this comparison should be read as indicative only.

c. Data-Quality Findings:

Preparing the data exposed inconsistencies that a query tool has to cope with. The internal-marks component is labelled INT and EXT in semesters IV and V but IA and SEE in semester VI. Some sheets carry a wrong semester or year label, and two of the workbooks contain an extra sheet named wob that repeats the section IV-A data. For semesters IV and V, the number of failures recorded in the student sheets differs from the department's summary sheets in five of six sections. For example, section IV-B shows 10 failures in the student sheet and 12 in the summary, and section V-B shows 4 against 7. The semester VI summary lists section strengths of 63, 69 and 69, whereas the student sheets contain 70, 72 and 61 students. Five failing entries in semester VI have no total or percentage recorded. The pass percentages in this paper use the status recorded in the student sheets.

d. Status of Agent Evaluation:

The example questions above form the beginning of a test set. For each, the correct SQL and its result are known. The next step is to give the same questions to the language model and compare its generated

queries and answers with these ground-truth results, using execution accuracy and answer correctness as measures. This comparison has not yet been carried out, so no accuracy figure is claimed for the agent in this paper.

8. References

- [1] T. Yu, R. Zhang, K. Yang, M. Yasunaga, D. Wang, Z. Li, J. Ma, I. Li, Q. Yao, S. Roman, Z. Zhang, and D. Radev, "Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task," in Proc. Conf. Empirical Methods in Natural Language Processing (EMNLP), Brussels, Belgium, 2018, pp. 3911–3921.
- [2] J. Li et al., "Can LLM already serve as a database interface? A BIG bench for large-scale database grounded text-to-SQLs," in Advances in Neural Information Processing Systems (NeurIPS), 2023.
- [3] M. Pourreza and D. Rafiei, "DIN-SQL: Decomposed in-context learning of text-to-SQL with self-correction," in Advances in Neural Information Processing Systems (NeurIPS), 2023.
- [4] D. Gao, H. Wang, Y. Li, X. Sun, Y. Qian, B. Ding, and J. Zhou, "Text-to-SQL empowered by large language models: A benchmark evaluation," Proc. VLDB Endowment, vol. 17, no. 5, 2024.
- [5] X. Chen, M. Lin, N. Schärli, and D. Zhou, "Teaching large language models to self-debug," in Proc. Int. Conf. Learning Representations (ICLR), 2024.
- [6] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "ReAct: Synergizing reasoning and acting in language models," in Proc. Int. Conf. Learning Representations (ICLR), 2023.