An LLM-Based Approach for Insight Generation in Data Analysis

Alberto Sánchez Pérez
Aily Labs, EURECOM
alberto.sanchez@ailylabs.com
sanchez@eurecom.fr

Alaa Boukhary
Aily Labs
alaa.boukhary@ailylabs.com

Paolo Papotti
EURECOM
papotti@eurecom.fr

Luis Castejón Lozano
Aily Labs
luis.castejon@ailylabs.com

Adam Elwood
Aily Labs
adam.elwood@ailylabs.com

Abstract

Generating insightful and actionable information from databases is critical in data analysis. This paper introduces a novel approach using Large Language Models (LLMs) to automatically generate textual insights. Given a multi-table database as input, our method leverages LLMs to produce concise, text-based insights that reflect interesting patterns in the tables. Our framework includes a Hypothesis Generator to formulate domain-relevant questions, a Query Agent to answer such questions by generating SQL queries against a database, and a Summarization module to verbalize the insights. The insights are evaluated for both correctness and subjective insightfulness using a hybrid model of human judgment and automated metrics. Experimental results on public and enterprise databases demonstrate that our approach generates more insightful insights than other approaches while maintaining correctness.

1 Introduction

In an era of data-driven decision-making, the ability to extract insights from complex databases has become increasingly important (Jahns, 2013; Steiner, 2022). These insights provide meaningful and actionable information derived from raw data. Insights are often derived through statistical, analytical, and computational methods. For example, given the database in Figure 1, a valuable insight would be *"Higher percentages of students eligible for free or reduced-price meals (FRPM) correlate with lower average SAT scores in reading, math, and writing."*

Insights are essential to make informed decisions. This is crucial in business contexts, healthcare, or scientific research, where better insights lead to more effective outcomes and advancements (Khan, 2024; Abdul-Azeez et al., 2024; Daraojimba et al., 2024).

Generating insights traditionally requires significant manual effort. Analysts must meticulously preprocess, explore, and refine the data. It also requires specialized knowledge of the domain to analyze and interpret the data, further complicating the task. This makes the process both time-consuming and resource-intensive (Arora and Malik, 2015; Bean, 2022).

There have been multiple proposals for automating data analysis tasks (Ma et al., 2023; Ding et al., 2019; OpenAI, 2024a; LangChain, 2024b). However, their outputs are not as insightful as human-crafted insights. They often only operate on single tables and require pre-cleaned data or user-defined goals, which limit their applicability to real-world use cases.

Our approach introduces a novel framework leveraging Large Language Models (LLMs) to automatically generate insights. However, the process of extracting valuable insights from raw data with LLMs is non-trivial. First, LLMs have limited context size and cannot be fed with entire databases (Pawar et al., 2024). Second, LLMs still struggle in processing structured data (Li et al., 2024a). Because of these issues, we adopt an agentic approach, where external tools (SQL scripts, in our case) are used by the LLM to handle structured data. While this is a popular solution, existing methods mostly produce "shallow" insights, such as the identification of simple trends or outliers (OpenAI, 2024a; LangChain, 2024b). As depicted in Figure 1, our idea is to obtain richer insights by first creating high-level questions over the database. Those questions are more complex than the insights that can be directly generated by the LLM. We exploit the fact that LLMs are able to split a question down into simpler ones that can

Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 562–582

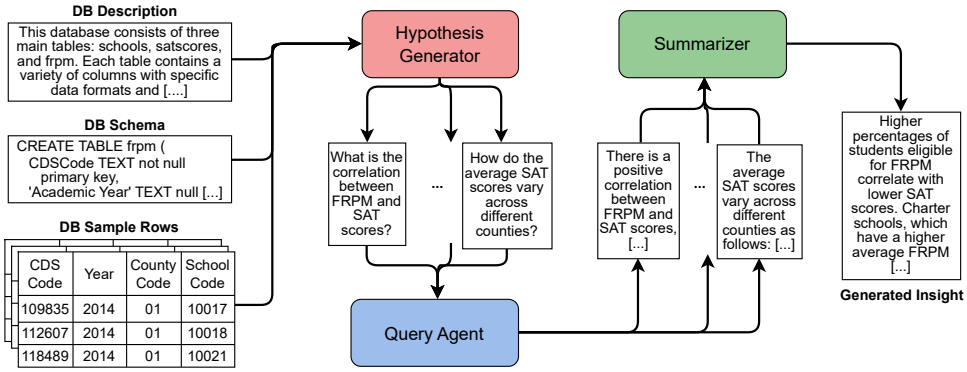

Figure 1: Overview of our approach: the Hypothesis Generator generates interesting, high-level questions, the Query Agent answers them using SQL scripts, and the Summarization module aggregates the results into an insight.

be turned into SQL scripts and validated over the database. Finally, the query results are aggregated into a textual insight.

The generated insights are evaluated using a hybrid approach that combines human judgment with automated metrics, ensuring a rigorous assessment of both correctness and insightfulness. Experiments on public and real company databases show that our solution clearly outperforms existing alternatives in terms of insightfulness of the output texts.

2 Preliminaries

We first introduce Text-to-SQL as it plays a key role in our solution. We then define insights and the criteria for evaluating them.

2.1 Text-to-SQL

A database D is defined as a set of tables T_i, each table is a set of tuples $T_i = (t_{j1}, t_{j2}, \ldots, t_{jn}) \mid j \in 1 \ldots m$ with j representing the columns of the table. A query q is a mapping between a list of input tables and an output table, such that $q : T_0 \times \ldots \times T_k \rightarrow T_O$, where q is a sequence of operations, denoted as $Q = f_n \circ \ldots \circ f_0$. Each f_i consists of relational operations (e.g., projection, selection, join), set operations (e.g., union, intersection), and set functions (e.g., sum, average). The result of applying query q on database D is denoted as $q(D) = R$, where R is the resulting table.

The objective of Text-to-SQL can be defined as: $\min dist(q_{gen}(D), q_{GT}(D))$, where q_{gen} is the query generated by the model, q_{GT} is the ground truth query, which answers the question, and *dist* is a distance function between the resulting tables.

2.2 Insights

We define an insight I as a short text derived from a database D. We set that it should not be longer than 3 sentences based on experiments with our product, which indicate that longer text is less effective in maintaining reader interest.

Insights are derived using information from a subset or derivation of the database $f(D)$. For example, $f(D)$ can be the result of one or more SQL queries over D or a transformation of D through code. In addition to $f(D)$, the input includes textual information *info*: $\{f(D)\} \cup info \rightarrow I$, with *info* = $\{D_{\text{info}}, D_{\text{schema}}\}$, i.e., D_{schema} contains the schema and a sample of rows of each table of D (DB Schema and DB Sample Rows in Figure 1); D_{info} is a textual high-level description of (i) the database, (ii) each table, (iii) the columns in each table (DB Description in Figure 1). If not available in the database catalog, This D_{info} is LLM-generated with a prompt (Appendix E.1), following the "verbose prompt" strategy (Sun et al., 2024).

2.2.1 Insightfulness

Insightfulness is a subjective metric of insights based on human judgment. It is hard to quantify objectively. This is due to insightfulness being case, domain, and user dependent. The metric is based on a set of implicit metrics $\{M_0(I, U), \ldots, M_n(I, U)\}$ (e.g., actionability, relevance, novelty, ...) driven by expertise and KPIs. These metrics cannot be explicitly calculated given only D as input and must be estimated by the user U, weighted by their preferences $\{w_{M_0}, \ldots, w_{M_n}\}$. We then define insightfulness

of an insight to a user as:

$$\textit{insightfulness}(I,U) = \frac{\sum_{i=0}^{n} w_{M_i} \cdot M_i(I,U)}{\sum_{i=0}^{n} w_{M_i}}$$

2.2.2 Correctness

Insights can be composed of one or more claims C_i. Claims are factual statements that have a truth value TV that can be proven true or false, thus $TV(C_i) \in \{0,1\}$. We then define the correctness of I as the mean of the truth value of each of its claims:

$$\textit{correctness}(I) = \frac{1}{n}\sum_{i=0}^{n} TV(C_i)$$

3 Problem Definition

The main objective of this work is the generation of short texts that are both *insightful* and *correct* (Sections 2.2.2 and 2.2.1).

Depending on the context, insightfulness and correctness might have different importance. We define the weighting factor α in the resulting function. The objective is to maximize the weighted harmonic mean:

$$O = \max \frac{1}{\frac{\alpha}{\textit{insightfulness}} + \frac{1-\alpha}{\textit{correctness}}}$$

By default, we assign insightfulness and correctness the same importance ($\alpha = 0.5$).

4 Proposed Architecture

We generate insights with a 3-step architecture, as in Figure 2, that (1) generates high-level questions and splits each of them into simpler subquestions that are easier to answer, (2) then answers and validates them using SQL to (3) finally post-process them into the final insight.

4.1 Hypothesis Generator

The Hypothesis Generator uses a multi-step approach to generate questions that are both interesting and easy to answer for the given database. It is composed of two elements. A high-level generator HL-G that uses a prompt (Appendix E.2) to generate overarching questions h_i. The high-level generator uses a short description of the database ($\textit{short}(D_{\text{info}})$) that we shorten to prevent constraining the model's response with database details, enabling more exploratory questions within the domain:

$$\text{HL-G}(\textit{short}(D_{\text{info}})) \rightarrow h_0, \ldots, h_n$$

A low-level generator LL-G splits the high-level question h_i into subquestions s_{ij}. By using specific prompting (Appendix E.2) and the full description and schema of the database as input, we constrain the model to formulate more precise and concrete questions, which are easier to answer in the next step:

$$\text{LL-G}(h_i, D_{\textit{info}}, D_{\textit{schema}}) \rightarrow s_{i0}, \ldots, s_{im}$$

4.2 Query Agent

Our work employs SQL queries instead of pandas to overcome the speed, storage, and scalability limitations of dataframes. The Query Agent *QAgent* generates a SQL query q_{ij} and resulting tables R_{ij} ($q_{ij}(D) = R_{ij}$) to answer the subquestions using the database information and schema:

$$\textit{QAgent}(s_{ij}, D_{\textit{info}}, D_{\textit{schema}}) \rightarrow \ q_{ij}$$

The objective of the *QAgent* is to minimize $\textit{dist}(q_{ij}(D), q^*_{ij}(D))$, where q^*_{ij} is the ground truth query and *dist* is a distance function between tables based on the cell metrics in (Papicchio et al., 2023). More concretely, we define *dist* as the harmonic mean of the metrics *cell-precision* and *cell-recall*, $\textit{dist} = \frac{2 \cdot \textit{cell-precision} \cdot \textit{cell-recall}}{\textit{cell-precision} + \textit{cell-recall}}$.

Queries are then verbalized $\textit{verb}(R_{ij})$ in natural language with a prompt (Appendix E.6) to answer s_{ij}. They are then validated using LLM evaluation functions that use an LLM score, $\textit{score}_a$ of answer relevance and answerability (Lin and Chen, 2023), to remove questions with a score under a threshold τ_a[1].

4.3 Summarization

The Summarization step summarizes (*summ*) the verbalized answers to generate the Insight:

$$\textit{summ}(\textit{verb}(R_{i0}), \ldots, \textit{verb}(R_{im})) \rightarrow I\ ;$$

Algorithm 1 Iterative Reflection Based on Verb Relations

```
i ← 0
while score_h(I, {verb(R_i0), ..., verb(R_im)}) ≥ τ_a and i < maxit do
    reflect(I, {verb(R_i0), ..., verb(R_im)}) → I
    i ← i + 1
end while
return I
```

[1]The experimentally determined value of τ_a is 0.7 for both answerability and relevance, effectively filtering most low-quality answers while retaining a sufficient number of relevant ones.

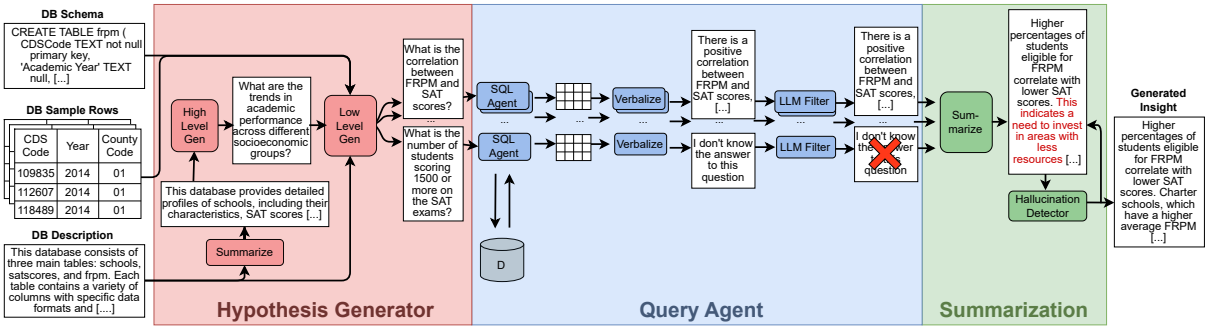

Figure 2: Proposed Architecture: The High-Level Generator generates questions using a short description , the Low-Level Generator splits each question into subquestions that are easier to answer by giving all the database details. The Query Agent uses SQL queries to answer those questions and validates them with LLM evaluation. Finally, the Summarizer aggregates the answers into a short insight and iteratively removes hallucinations to generate the final result.

Each generated insight is post-processed to filter out possible hallucinations, as detailed in Algorithm 1. An evaluation function $score_h$ is used to measure an LLM hallucination score by splitting the insight into different claims and using a LLM to generate a score based on the contradictions between them and the answers $verb(R_{ij})$ (Liu et al., 2023). The summary uses reflexion (Shinn et al., 2023) to iteratively correct the summary until the LLM hallucination score is under a threshold τ_h[2] or exceeds an iteration limit *maxit*.

5 Evaluation

Our proposed framework was evaluated on both *insightfulness* (Section 2.2.1) and *correctness* (Section 2.2.2) using a combination of human and LLM evaluation.

5.1 Experimental Setup

Datasets. We generated a *private dataset* from the company's internal data. This is used to evaluate the approach in real-world, proprietary contexts. It also allows for in-depth evaluation using domain experts from the company.

- **private_sales:** internal sales and sales forecasting data; it has 3 tables with 16-49 columns (median of 17).

We also employ a set of public datasets to ensure reproducibility and for a broader evaluation in a range of domains. These datasets a sampled from the BIRD benchmark (Li et al., 2023), which is composed from real data sources spanning 37 domains (Volvovsky et al., 2024).

We focus on a subset rather than the entire database benchmark to manage costs, as the *correctness* evaluation requires specialized human judgment. In the sampling process, we ensured thematic and structural variety across the databases by including:

- **california_schools:** education data in the state of California; it has 3 tables with 11-49 columns (median of 29).
- **codebase_community:** data from posts in the social media platform Reddit; it has 8 tables with 4-21 columns (median of 6).
- **debit_card_specializing:** card transactions data in the fuel industry; it has 5 tables with 2-9 columns (median of 3).
- **european_football_2:** football data with teams, players, and matches; it has 7 tables with 2-115 columns (median of 7).
- **student_club:** data from a student club with financial data, attendance metrics and events; it has 8 tables with 2-9 columns (median of 6).

LLMs. All methods use the same base model, GPT4o (OpenAI, 2024b), to have a fair comparison between them. We use GPT4o because of its effectiveness in generating structured outputs with Function Calling (OpenAI, 2024). The average cost of generating an insight with our method amounts to 63 cents, which significantly reduces the cost from manual insight generation of $140,

[2]The value of τ_h has been experimentally determined at 0.9 to ensure rigorous hallucination removal.

i.e., the average estimated cost at our own company for a human-crafted insight (see Appendix G for more details). Also, potential decreases in LLM processing costs in the future will further reduce the cost of the system.

For the Text-to-SQL model, we adopt the LangGraph Sql Agent (LangChain, 2024a). Following best practices to generate the queries (Rajkumar et al., 2022), the schema and 3 sample rows are given to the model.

Baselines and Models. We compare our solution against a set of baselines.

- **GPT-DA:** Insights are generated using the ChatGPT data analysis functionality with the prompt detailed in Appendix E.4. It takes as input a CSV file and uses pandas code generation to output insights (OpenAI, 2024a).
- **Quick:** Insights are generated using Quick-Insights from MS PowerBI (Microsoft). Results are obtained from pre-joined if possible. If not, from random individual tables (Ding et al., 2019).
- **Serial:** Insights are generated by serializing a subset of the database into an HTML table due to context windows limitations in the LLMs (Pawar et al., 2024), which is then fed into the LLM prompt detailed in Appendix E.4.

We also evaluate variations of our architecture described in Section 4 as an ablation study.

- **High-Level Insights** (HLI)**:** Insights are generated with high-level and low-level questions as described in Section 4.
- **High-Level Insights without Summarization of the description** (HLI-wS)**:** Same architecture as the HLI method, but the High-Level Generator uses the full database description as an input, instead of a short summary.
- **High-Level Insights without High-Level** (HLI-wH): Insights are generated directly with low-level questions, without high-level questions.

As detailed in Appendix B.2, the number of generated insights varies across different methods, influenced by characteristics of each method, like filtering or data preprocessing. Some methods like HLI, HLI-wH or HLI-wS have stricter filtering, leading to fewer total insights (30-40 in total), while existing methods have a slightly higher number of insights (40-50).

5.2 Insightfulness Evaluation

Due to the difficulty of measuring *insightfulness*(I, U), we use a relative measure to better estimate it. We define a ranker $R(U, I_A, I_B)$ that classifies the insight with the most insightfulness such that:

$$R(U, I_A, I_B) = \begin{cases} I_A & \text{if } \textit{insightfulness}(I_A, U) > \\ & \quad \textit{insightfulness}(I_B, U) \\ I_B & \text{otherwise} \end{cases}$$

We then define the *insightfulness* based on an Elo rating system as follows:

$$\textit{insightfulness}(I, U) = \\ \text{Elo}(I \mid R_0(U, I_{A0}, I_{B0}), \ldots, R_k(U, I_{Ak}, I_{Bk}))\ ;$$

Where the Elo rating of I is based on a list of pair-wise insight comparisons from the ranker $R_0(U, I_{A0}, I_{B0}), \ldots, R_k(U, I_{Ak}, I_{Bk})$. For example, the insight "Higher percentages of students eligible for free or reduced-price meals (FRPM) correlate with lower average SAT scores in reading, math, and writing." would get a better score than "Los Angeles County has the highest number of students eligible for free meals, totaling 898,610 students.".

All compared insights are sampled from a uniform distribution:

$$I_{A0}, I_{B0}, \ldots, I_{Ak}, I_{Bk} \sim \text{Uniform}(\{I_0, \ldots, I_n\})$$

The Elo score has proven useful in translating subjective pairwise comparisons into numeric scores to rank LLM performance (Chiang et al., 2024; Askell et al., 2021; Boubdir et al., 2023). After each insight comparison, the ranking R for each of the models A is updated in relation to the other model B based on the result of the comparison ($S_A = 1$ if $R(U, I_A, I_B) = I_A$; $S_A = 0$ else), using the formula

$$R'_A = R_A + K \cdot (S_A - E_A)$$

where E_A is the expected score of the model according to the formula:

$$E_A = \frac{1}{1 + 10^{(R_B - R_A)/400}}$$

In our implementation, we assigned each model an initial rating of 1000 and a k-factor K of 4 to stabilize the Elo ratings and avoid biases towards recent games (LMSys, May 22, 2023)[3]. However, Figure 3, 4 and 5 represent 95% confidence intervals with bootstrapping using different order of the comparisons. In those cases, we use a k-factor of 8 for better visualization to get slightly bigger intervals, as it has minimal effects on the final ordering after 100 comparisons, more details reported in Appendix D.

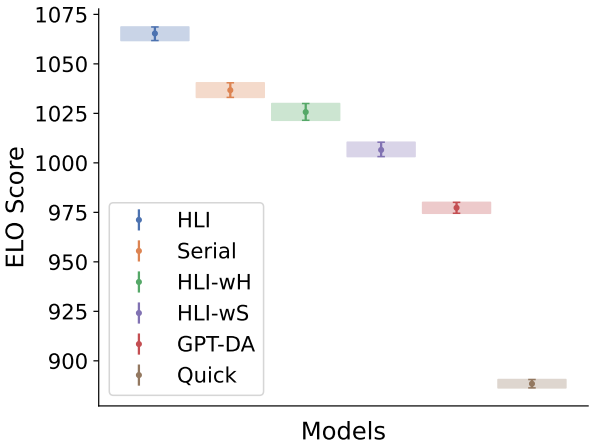

Figure 3: **Human** Elo Bootstrap Confidence Intervals (95%) in the database private_sales. Each bootstrap sample has a different order of comparisons.

Human Evaluation. For the human evaluation, we tasked a domain expert with performing 100 comparisons between insights generated by different methods on the private dataset. The human evaluation for correctness and insightfulness metrics was conducted separately, with a domain expert dedicating approximately one hour to evaluate insightfulness. As shown in Figure 3, our method, HLI, achieves the best results in terms of insightfulness. This comparison also works as an ablation test, showing the importance of the high-level and low-level question split (HLI-wH), as the split allows for deeper high-level questions that are answered through different perspectives with each subquestion. We also show that giving less information to the high-level generator (HLI-wS) allows for better *insightfulness*, as the high-level questions are less constrained due to not knowing the details of the database, allowing for more exploratory questions.

LLM Evaluation. To scale the evaluation of different insight generation models over a range of domains, we propose a new LLM-based evaluation model (the prompt for this task is reported in Appendix E.8).

Metric	Eval 1	Eval 2	Eval 3	Eval 4	LLM Eval
F1 score	0.750	0.774	0.702	0.748	**0.778**
Pearson	0.970	0.975	0.945	0.990	**0.997**

Table 1: Comparison of similarity metrics across results of the human (Eval 1, Eval 2, ...) and LLM (LLM eval) evaluators against the results of the domain expert. The table shows the F1 scores (across individual insight comparisons) and Pearson correlation coefficients (of the final ELO score of all models) between evaluators and domain expert on the private dataset.

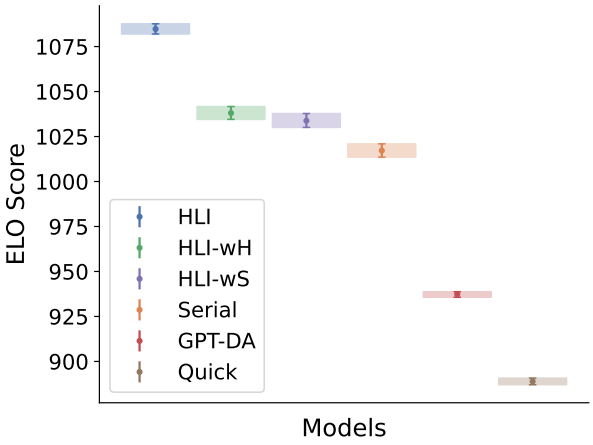

Figure 4: **LLM** Elo Bootstrap Confidence Intervals (95%) in the database private_sales. Each bootstrap sample has a different order of comparisons.

To evaluate the capability of the LLM-based model, we also tasked four non-domain experts to solve the same task of comparing the same 100 insights of the private_sales database. Then, the LLM evaluator (using the prompt from the Appendix E.8) was tasked with the same objective, with the results in Figure 4. As seen in Table 1, the LLM evaluator reports better similarity with the domain expert both in the insight-to-insight F1 score and in the end-to-end Pearson correlation between Elo scores. Higher scores and correlations suggest better agreement with the domain expert, thus the LLM evaluator can be used reliably to assess insightfulness.

[3]Intuitively, the rating differential between two LLMs acts as a predictor for the comparison outcome. For two LLMs with identical ratings, they are expected to achieve an equal number of victories. If an LLM's rating exceeds its opponent's by 100 points, the first LLM is predicted to win 64% of the time. If the rating differential is 200 points, the expected winning percentage for the higher-rated LLM rises to 76%.

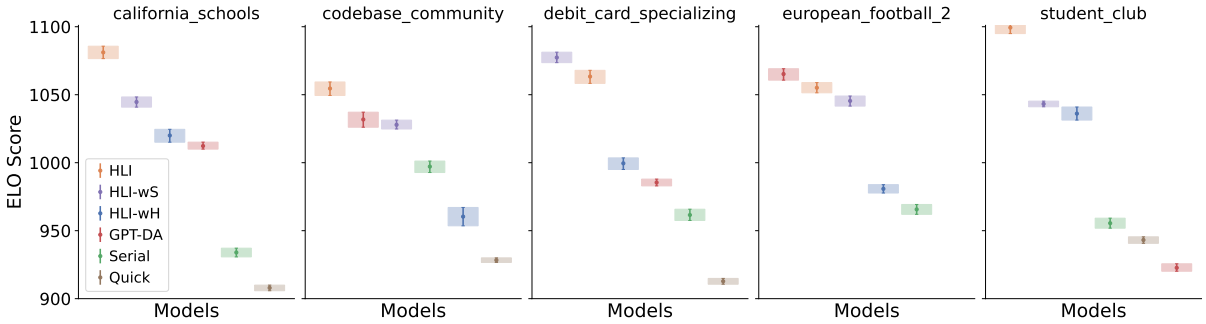

Figure 5: **LLM** Bootstrap Confidence Intervals (95%) in public BIRD databases. Each bootstrap sample has a different order of comparisons.

Figure 5 shows the results of the LLM evaluation of insightfulness on all public databases. Overall, all methods based on our approach perform better than baselines, with HLI clearly at the top and Quick performing the worst.

5.3 Correctness Evaluation

Following the correctness definition from Section 2.2.2, the truth value of each insight is the mean of the truth value of its claims scoring 1 when true and 0 when false. Some claims that contain multiple subclaims have been evaluated as $\frac{a}{b}$, where b is the number of subclaims and a, the number of correct ones.[4] We consider the score of each method to be the mean correctness of the evaluated insights from that method.

Human Evaluation. To evaluate correctness in the generated insights, a sample of 5 insights per database was hand-evaluated by the authors for each method (a total of 321 claims).

Figure 6 illustrates the correctness evaluation across different databases for every method. Quick gets perfect correctness across all databases due to its design. HLI-wS and HLI show slightly lower results, but still consistently correct overall; HLI-wH has a slightly lower correctness. Finally, Serial and GPT-DA get the lowest scores, with high variability. Overall, method HLI-wH gives the best tradeoff between *correctness* and *insightfulness* over most databases. This can be seen in Figure 7, which represents the performance of each model on a database as a point in a plot, where the x-axis corresponds to the *insightfulness* and the y-axis to *correctness*. The methods are then represented as polygons, with a point representing

[4]For example, the claim "The top 5 products are Nafta, Natural, Diesel, Myt Vozidel, and Diesel" has a truth value of $\frac{a}{5}$, where a is the number of correct products.

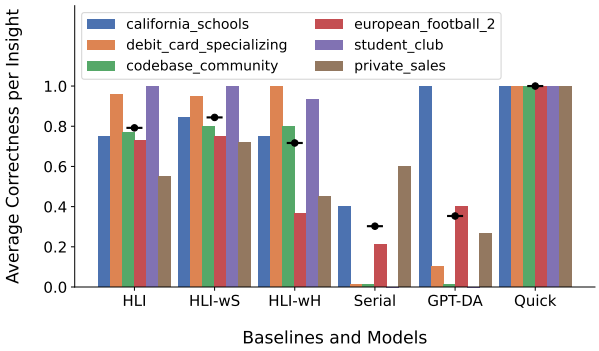

Figure 6: Average correctness per insight of all models across all databases. All evaluations with a total score of 0 correctness are due to having insights that have no truth value in relation to the data. For example an insight like the following: "Identifying users who frequently switch roles can help in creating targeted engagement strategies." suggests potential action that cannot be evaluated with data from the database.

the average performance, and an area representing the variability of its performance on different databases. Both HLI and HLI-wS have consistent results on the top-right part of the diagram, suggesting better performance, with HLI being more insightful overall and HLI-wS being slightly more correct. This aligns with the hypothesis that giving more information to the High-Level generator (as in HLI-wS) constraints the space of possibilities, making questions that are easier to answer but less insightful.

Additionally, a manual review of 58 randomly sampled summaries and answers generated by our method revealed no hallucinations by the LLM when summarizing or transcribing the results of the generated SQL queries. The most common source of errors stemmed from semantic parsing issues.

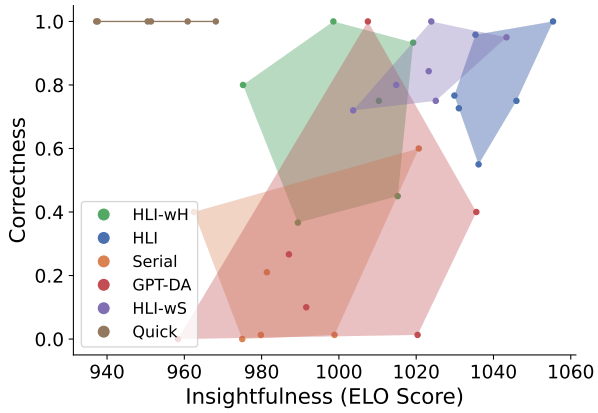

Figure 7: Dual evaluation of insightfulness (x-axis) and correctness (y-axis). Each polygon represents the performance variability of a model across different databases, with each point corresponding to the evaluation of one database. Larger polygon areas suggest greater variability across different domains and databases.

6 Related Work

We first position our work w.r.t. the literature on insight generation. We then discuss Text-to-SQL and how it is relevant to our solution.

Automatic Insights. Several models have been proposed for insight generation from tabular data, such as InsightPilot (Ma et al., 2023), QuickInsights (Ding et al., 2019), OpenAI Data Analysis (OpenAI, 2024a), or Langchain's Pandas Agent (LangChain, 2024b).

These works propose different definitions of insights. Some approaches consider insights derived from predefined templates (Law et al., 2020) populated with aggregate measures across subsets of tables. The generation of these insights is done either with a mining framework based on metrics (Ding et al., 2019) or using an LLM to guide the process (Ma et al., 2023). Although this approach offers several strengths, including providing visual data and ensuring factual correctness, it also presents certain limitations. Other approaches (OpenAI, 2024a; LangChain, 2024b) directly leverage LLMs to generate code to obtain insights. These methods are often specialized in single-step data analysis tasks (for example, calculating a correlation or doing some regression task) based on very concrete user instructions. All previous approaches are based on one or more of the following assumptions and limitations:

- Some approaches have only been tested on simple table structures and have not been evaluated on multi-table databases (Ma et al., 2023; Ding et al., 2019; OpenAI, 2024a; LangChain, 2024b).
- Some approaches require a user-defined goal (for example, "Show me interesting trends in mathematics scores for students") to guide the insight generation process (Ma et al., 2023; Ding et al., 2019; OpenAI, 2024a; LangChain, 2024b).
- Some approaches are template-based and require clean data that has been filtered and descriptive naming in columns (Ding et al., 2019). For example, the insight "[Subject]=[Math] has an increasing trend over [Time]" becomes meaningless if the placeholders lack descriptive names or if they are irrelevant to the context, e.g., finding trends on IDs or telephone numbers.

Text-to-SQL. Text-to-SQL is one of the most popular approaches for enabling natural language interfaces (NL) to databases. It consists of converting NL questions into valid SQL queries, where the result of the query is a set of tuples that answer the question.

The advent of LLMs has facilitated the development of complex architectures, such as LLM chains and agentic flows, which have significantly enhanced the efficacy of Text-to-SQL systems (Tai et al., 2023; Li et al., 2024c; Gao et al., 2023; Pourreza and Rafiei, 2023). Our work employs such architectures to facilitate the automatic generation of insights from databases via automatically generated SQL queries. While there is evidence that Text-to-SQL systems struggle with proprietary, large schemas (Chen et al., 2024; Papicchio et al., 2023), there are promising solutions to tackle this problem with prompt compression (Li et al., 2024b; Corallo and Papotti, 2024).

7 Conclusions

We have introduced a new method to generate insights from multi-table databases. We proposed an architecture that generates interesting questions, breaks them into more easily answerable subquestions, answers them with automatic SQL queries, and summarizes all results into a final insight. We evaluated the insightfulness and correctness of our method using human and LLM evaluation against

three baselines and two variations. We demonstrated that our method achieves better and more consistent results on these two metrics.

For future research, the model's evaluation could be extended to encompass a broader range of domains or data sources (e.g., textual documentation using RAG, Knowledge Graphs or Internet search). Also, there are techniques that might enhance the performance, such as k-shot learning or prompt learning techniques (Yuksekgonul et al., 2024). Finally, an interactive exploration of the insight search space could lead to better outputs with a limited effort, for example by extending the solution to consider user feedback on early results.

8 Limitations

Generated insights are not always perfectly correct, which in some contexts may lead to unexpected decisions. This effect can be alleviated by reducing database complexity or with future models that have better results on the Text-to-SQL task. Additionally, the architecture of the model involves a lot of LLM calls, which can have downsides in terms of cost (see Appendix G for more details). Although different domains have been evaluated, our effort is still limited by cost and some domains have not been covered. Finally, only English insights have been generated and our architecture has only been evaluated with one LLM.

9 Ethical Considerations

The use of LLMs for generating automatic data insights presents several ethical challenges.

A primary concern in all works that use LLMs is the potential risk of biases, as large datasets may inadvertently reinforce societal prejudices. We also need to account for its environmental impact, due to the high computational demands of LLMs.

There is also a risk of misuse of our system, as generated insights with our framework may be exploited for unethical purposes, such as manipulation or surveillance.

Another risk of our system is its implementation in real-life scenarios. Automated systems should not be used to deflect responsibility in decision-making contexts, they should be used as a supporting tool. Finally, there is a risk of job displacement, as reliance on AI may reduce the need for human analysts in the long term.

10 Acknowledgements

We thank the action editor and the reviewers for their comments that helped us improve the content of our work. We also thank all the people from Aily who have helped us, especially Roumila Chebil and the genAI team (Gerard Conangla, Nekane Guarrotxena, Marco Introvigne, Miguel Conner, Fernando Rodríguez, Juan Abia and Joshua Fellowes).

References

Oluwatosin Abdul-Azeez, Alexsandra Ogadimma Ihechere, and Courage Idemudia. 2024. Enhancing business performance: The role of data-driven analytics in strategic decision-making. *International journal of management and entrepreneurship research*, 6(7):2066–2081.

Deepali Arora and Piyush Malik. 2015. Analytics: Key to go from generating big data to deriving business value.

Amanda Askell, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, Andy Jones, Nicholas Joseph, Ben Mann, Nova DasSarma, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Jackson Kernion, Kamal Ndousse, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, and Jared Kaplan. 2021. A general language assistant as a laboratory for alignment.

Randy Bean. 2022. Why becoming a data-driven organization is so hard. *Harvard Business Review*. Available online: https://hbr.org/2022/02/why-becoming-a-data-driven-organization-is-so-hard.

Meriem Boubdir, Edward Kim, Beyza Ermis, Sara Hooker, and Marzieh Fadaee. 2023. Elo uncovered: Robustness and best practices in language model evaluation.

Peter Baile Chen, Fabian Wenz, Yi Zhang, Moe Kayali, Nesime Tatbul, Michael J. Cafarella, Çagatay Demiralp, and Michael Stonebraker. 2024. BEAVER: an enterprise benchmark for text-to-sql. *CoRR*, abs/2409.02038.

Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Hao Zhang, Banghua Zhu, Michael Jordan, Joseph E. Gonzalez, and Ion Stoica. 2024. Chatbot arena: An open platform for evaluating llms by human preference.

Giulio Corallo and Paolo Papotti. 2024. Finch: Prompt-guided key-value cache compression for large language models. *Transactions of the Association for Computational Linguistics*, 12:1517–1532.

Creative Commons. 2013. Attribution-sharealike 4.0 international (cc by-sa 4.0). https:

//creativecommons.org/licenses/by-sa/4.0/.

Andrew Ifesinachi Daraojimba, Chidera Victoria, Ibeh, Oluwafunmi Adijat Elufioye, Temidayo Olorunsogo, Onyeka Franca Asuzu, and Ndubuisi Leonard Ndubuisi. 2024. Data analytics in healthcare: A review of patient-centric approaches and healthcare delivery. *World Journal Of Advanced Research and Reviews*.

Rui Ding, Shi Han, Yong Xu, Haidong Zhang, and Dongmei Zhang. 2019. Quickinsights: Quick and automatic discovery of insights from multi-dimensional data. In *Proceedings of the 2019 ACM International Conference on Management of Data (SIGMOD'19 industrial track)*.

Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding, and Jingren Zhou. 2023. Text-to-sql empowered by large language models: A benchmark evaluation.

Veit Jahns. 2013. Data insights: new ways to visualize and make sense of data by hunter whitney. *ACM Sigsoft Software Engineering Notes*, 38(6):45–46.

Attia Khan. 2024. Effective decision making using data analytics. *Indian Scientific Journal Of Research In Engineering And Management*, 08(04):1–5.

LangChain. 2024a. Langgraph sql agent. https://langchain-ai.github.io/langgraph/tutorials/sql-agent/.

LangChain. 2024b. Pandas dataframe. https://python.langchain.com/v0.2/docs/integrations/toolkits/pandas/.

Po-Ming Law, Alex Endert, and John T. Stasko. 2020. Characterizing automated data insights. *CoRR*, abs/2008.13060.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Rongyu Cao, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin C. C. Chang, Fei Huang, Reynold Cheng, and Yongbin Li. 2023. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls.

Peng Li, Yeye He, Dror Yashar, Weiwei Cui, Song Ge, Haidong Zhang, Danielle Rifinski Fainman, Dongmei Zhang, and Surajit Chaudhuri. 2024a. Table-gpt: Table fine-tuned gpt for diverse table tasks. *Proc. ACM Manag. Data*, 2(3).

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. 2024b. SnapKV: LLM knows what you are looking for before generation. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Zhishuai Li, Xiang Wang, Jingjing Zhao, Sun Yang, Guoqing Du, Xiaoru Hu, Bin Zhang, Yuxiao Ye, Ziyue Li, Rui Zhao, and Hangyu Mao. 2024c. Pet-sql: A prompt-enhanced two-stage text-to-sql framework with cross-consistency.

Yen-Ting Lin and Yun-Nung Chen. 2023. Llm-eval: Unified multi-dimensional automatic evaluation for open-domain conversations with large language models.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. G-eval: Nlg evaluation using gpt-4 with better human alignment.

LMSys. May 22, 2023. Chatbot arena: Elo rating calculation. https://colab.research.google.com/drive/17L9uCiAivzWfzOxo2Tb9RMauT7vS6nVU?usp=sharing.

Pingchuan Ma, Rui Ding, Shuai Wang, Shi Han, and Dongmei Zhang. 2023. Demonstration of insightpilot: An llm-empowered automated data exploration system.

Microsoft. Power bi. https://powerbi.microsoft.com.

OpenAI. 2024a. Data analysis with chatgpt. https://help.openai.com/en/articles/8437071-data-analysis-with-chatgpt.

OpenAI. 2024b. Hello gpt-4o. https://openai.com/index/hello-gpt-4o/.

OpenAI. 2024. Openai function calling. https://platform.openai.com/docs/guides/function-calling.

Simone Papicchio, Paolo Papotti, and Luca Cagliero. 2023. QATCH: Benchmarking SQL-centric tasks with table representation learning models on your data. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Saurav Pawar, S. M Towhidul Islam Tonmoy, S M Mehedi Zaman, Vinija Jain, Aman Chadha, and Amitava Das. 2024. The what, why, and how of context length extension techniques in large language models – a detailed survey.

Mohammadreza Pourreza and Davood Rafiei. 2023. Din-sql: Decomposed in-context learning of text-to-sql with self-correction.

Nitarshan Rajkumar, Raymond Li, and Dzmitry Bahdanau. 2022. Evaluating the text-to-sql capabilities of large language models.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning.

Stefan Steiner. 2022. Harnessing data to make better-informed decisions. *Scientia*.

Ruoxi Sun, Sercan Ö. Arik, Alex Muzio, Lesly Miculicich, Satya Gundabathula, Pengcheng Yin, Hanjun Dai, Hootan Nakhost, Rajarishi Sinha, Zifeng Wang, and Tomas Pfister. 2024. Sql-palm: Improved large language model adaptation for text-to-sql (extended).

Chang-Yu Tai, Ziru Chen, Tianshu Zhang, Xiang Deng, and Huan Sun. 2023. Exploring chain of thought style prompting for text-to-SQL. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5376–5393, Singapore. Association for Computational Linguistics.

Shai Volvovsky, Marco Marcassa, and Mustafa Panbiharwala. 2024. Dfin-sql: Integrating focused schema with din-sql for superior accuracy in large-scale databases.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. 2024. Textgrad: Automatic "differentiation" via text.

A Example of an Insight

Generated by High-Level Insights

Insight: Higher percentages of students eligible for free or reduced-price meals (FRPM) correlate with lower average SAT scores in reading, math, and writing. Charter schools, which have a higher average FRPM eligibility (61.07%), also show lower SAT scores compared to non-charter schools. This indicates a strong link between socioeconomic status and academic performance.

High-Level Question: What are the trends in academic performance across different socioeconomic groups?

Subquestions:

- Are there significant differences in SAT performance between charter schools and non-charter schools, particularly in relation to their students' socioeconomic status?
- How do the average SAT scores in reading, math, and writing vary across different counties?
- For schools with a high percentage of students eligible for free or reduced-price meals (>70%), what is the average number of students scoring 1500 or above on the SAT?
- What is the correlation between the percentage of students eligible for free or reduced-price meals and the average SAT scores in reading, math, and writing?

B Dataset Details

B.1 Dataset License

The generated dataset of insights is licensed under CC BY-SA 4.0 (Creative Commons, 2013). This dataset is intended for research purposes.

The released dataset does not contain any personal or confidential information and is fully derived from public data (Li et al., 2023). All generated insights are in English.

B.2 Number of insights

Model	HLI	HLI-wS	HLI-wH	Serial	GPT-DA	Quick
california_schools	7	9	10	9	9	10
codebase_community	9	6	9	9	10	6
debit_card_specializing	7	5	10	9	10	10
european_football_2	7	5	6	9	10	10
private_sales	6	6	7	9	10	9
student_club	6	6	10	9	10	10

Table 2: Number of insights generated per database for each method.

Table 2 shows that the number of generated insights varies from method to method because of different reasons:

- HLI, HLI-wS and HLI-wH: Because of the LLM evaluation functions which filter final insights if they get a low score.
- GPT-DA: In one instance the model only generated 9 insights even when explicitly it was told to generate 10.
- Quick: The model was either impossible to pre-join (private_sales) or the pre-joined table only generated 1 number (codebase_community). In both cases, insights were drawn from randomly sampled individual tables.

B.3 Number of subquestions

Model	HLI	HLI-wS	HLI-wH	Serial	GPT-DA	Quick
california_ schools	3.71	4.78	1.00	1.00	1.00	1.00
codebase_ community	3.89	1.83	1.00	1.00	1.00	1.00
debit_card_ specializing	4.00	2.20	1.00	1.00	1.00	1.00
european_ football_2	2.14	3.00	1.00	1.00	1.00	1.00
private_ sales	2.00	1.83	1.00	1.00	1.00	1.00
student_ club	3.00	3.50	1.00	1.00	1.00	1.00

Table 3: Average number of subquestions generated for each insight

The average number of subquestions generated by each insight is presented in Table 3.

B.4 Insight length

Model	HLI	HLI-wS	HLI-wH	Serial	GPT-DA	Quick
california_ schools	424.0	412.0	270.2	207.89	226.78	89.0
codebase_ community	349.11	298.5	177.78	271.44	199.0	87.0
debit_card_ specializing	438.86	332.8	213.6	246.89	218.7	104.6
european_ football_2	297.57	323.4	217.33	307.78	211.1	60.9
private_ sales	423.83	314.0	242.29	350.33	99.0	90.56
student_ club	270.0	257.83	118.10	296.22	471.67	55.25

Table 4: Average number of characters per insight

The average length in characters for each insight is presented in Table 4.

C Evaluation Details

C.1 Instructions

Instructions for the evaluators

You will receive a list containing pairs of insights. Your task is to decide which insight in each pair is more insightful based on the provided insightfulness criteria:

Insightfulness measures the quality of an insight. More concretely, it measures criteria like

- How impactful/actionable the insight is, if it can lead to any key actions and what the impact of those actions would be (ideally over measurable KPIs)
- How interesting and relevant is the information provided.
- How clear, precise and coherent the insight is, how easily and accurately can you understand the information it provides
- ...

All these criteria are subjective and may vary based on your expertise, domain, and specific interests. You'll need to use your judgement to assess which insight offers more value based on these criteria.

Evaluation Process

1. Read both insights
2. Compare the insights and choose the more insightful one
3. Write your chosen insight in the Best_Insight column. Write "1" if you choose the insight to the left and "2" if you choose the one to the right

Additional Guidelines

- Correctness: The correctness of insights will be evaluated separately. Do not factor correctness into your evaluation.
- Length Variability: Insights may vary in length.

C.2 Evaluators Profile

C.2.1 Recruitment

All evaluators were recruited from our company.

C.2.2 Data Statement

The following is a self-reported data statement.

D ELO K-Factor Study

Figure 8 shows a study on the effect of the k-factor hyperparameter on the ELO evaluation of our domain expert. We can see that higher k-factors pro-

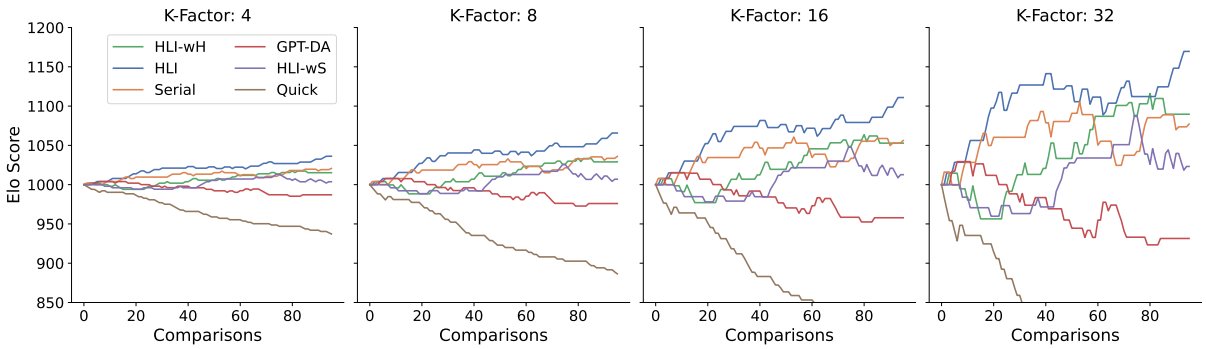

Figure 8: Study on the effect of the k-factor on the ELO evolution. We can see that 100 comparisons are enough to grant convergence and stability. However, greater k-factors like 16 or 32 give more weight to the latest comparisons (giving a better last ELO score to HLI-wH over Serial just for winning the latest comparisons).

Evaluator	Age	Gender	Nationality	Professional Background
Expert Eval	40-50	Female	Algeria/ Germany	Business Impact Lead
Eval 1	30-40	Male	Spain	AI Engineer
Eval 2	20-30	Female	Spain	AI Engineer
Eval 3	20-30	Male	Italy	AI Engineer
Eval 4	20-30	Male	USA	AI Engineer

Table 5: Summary of Evaluators

vide faster but less stable convergence, with more weight of the latest comparisons. With a k-factor of 4 or 8, our study suggests 100 comparisons are enough to converge into a final result. We can see that a k-factor of 16 or 32 gives different final ELO results (HLI-wH gets a better position than GPT-DA) due to the increased weight of latest comparisons.

E Prompts

E.1 Database Description

Prompt 1: Database Description Creation Prompt

Given the following database create a description of the database in natural text explaining to a user the structure of the table and the data it contains. Avoid html and give raw text with all the explanations. Explain the database, the tables and the columns in each table.
{db_schema}
{db_sample_rows}

E.2 High-Level Insights Prompts

Prompt 2: High-Level Generator Prompt

Generate ten questions related to a user that works as an analyst from the following data sources: given a retail worker who has to analyze customer data - what are the customer trends, and what actions might they need to take using data sources. The user works with the following database: {tables_description}

Prompt 3: Low-level Generator Prompt

We have a platform for users that need to analyse data.
We have access to these sql tables {tables}. The tables have the following description {tables_description}. We have the following complex questions {questions}.
Generate subquestions that a user can use to answer the complex question which might be answered from these tables. The questions should be answered in the form of insights that can be used to make decisions not just information about some numbers.The questions should be very verbose that if sum is needed say sum and if average is needed say average of certain columns. The questions should non sequential and can be executed in parallel. You can use the following schema as a reference: {schema}

E.3 High-Level Insight without High-Level (HLI-wH)

Prompt 4: Hypothesis Generator prompt

We have a platform for users that need to analyse data.
We have access to these sql tables {tables}.
The tables have the following description {tables_description}.
Generate ten questions related to customer consumption data in the industry for the user from the following data sources: The questions should be answered in the form of insights that can be used to make decisions not just information about some numbers.
The questions should be very verbose that if sum is needed say sum and if average is needed say average of certain columns. The questions should be non sequential and can be executed in parallel. You can use the following schema as a reference: {schema}

E.4 Baselines

Prompt 4: Hypothesis Generator prompt

We have a platform for users that need to analyse data.
We have access to these sql tables {tables}.
The tables have the following description {tables_description}.
Generate ten insights (3 lines long) related to customer consumption data in the industry for the user from the following data sources: The insights must be used to make decisions not just information about some numbers.
You can use the following schema as a reference: {schema}
Data: {Data}

E.5 Short Database Description

Prompt 5: Hypothesis Generator prompt

Take the following database description and return a brief (3 lines) description.
Your description must contain only a high-level description of the database, avoid explaining tables.
###
Database description: {tables_description}

E.6 SQL Agent

The following prompt is from the langgraph sql agent: (LangChain, 2024a)

Prompt 6: SQL Agent Prompt from Langgraph

You are an agent designed to interact with a SQL database.
Given an input question, create a syntactically correct {dialect} query to run, then look at the results of the query and return the answer
Unless the user specifies a specific number of examples they wish to obtain, always limit your query to at most {top_k} results.
You can order the results by a relevant column to return the most interesting examples in the database.
Never query for all the columns from a specific table, only ask for the relevant columns given the question.
You have access to tools for interacting with the database.
Only use the below tools. Only use the information returned by the below tools to construct your final answer.
You MUST double check your query before executing it. If you get an error while executing a query, rewrite the query and try again.
DO NOT make any DML statements (INSERT, UPDATE, DELETE, DROP etc.) to the database.
If the question does not seem related to the database, just return "I don't know" as the answer.

E.7 Summarization Prompts

Prompt 7: Summarizer prompt

As an AI programmed to simulate an expert-level business analyst, your task is to construct a short strategic business insight from provided data.

Could you please provide a concise and comprehensive summary of the given text? The summary should capture the main points and key details of the text while conveying the author's intended meaning

accurately. The length of the summary should be around 3 lines, gathering the main points and representing it in 3 lines.

The final insight should seem realistic and actionable, serving as a powerful tool for decision-makers to visualize potential strategies and outcomes.

Your summary must follow this guidelines

- Don't write insights longer than 3 lines long.
- Avoid adding information or recommendations that don't come from the context.
- Avoid enumeration of facts; reason the high-level pattern from the data and most important information.
- Mention the important data to support the inferred high-level patterns.
- Avoid focusing on individual entities, unless it's for exemplifying a pattern.
- Provide information that is actionable, not just random facts; it must only provide the interesting information.
- Avoid mentioning unimportant information.
- You don't need to mention all the information from the data, only the important one.

Remember to follow this instructions (after doing the summary, recheck that your summary followings them) or I will lose my job :(

High-Level Question
{hlquestion}

Context
{low_level_answers}

Prompt 8: Summarization with reflection prompt

Please reflect on your recent summarization task. You will be provided with the context and corresponding summary, alongside with the score that the summary received and its reasoning. You will need to produce a new summarized strategic business insight from provided data, considering the score and the reasoning to improve the result. Remember that the final insight should seem realistic and actionable, serving as a powerful tool for decision-makers to visualize potential strategies and outcomes.

Avoid adding extra information or recommendations. Focus on the data coming from the context, but don't just enumerate it, reason and provide an actionable insight for the user with the given task using the information from the context.

###

Context:
{low_level_answers}

###

Summary:
{summary}

###

Score:
This summary was evaluated with a score of: {score}

Reasoning:
The reasons for the score: {reasoning}

###

Guidelines for a Good Summary:

- Identify Key Points: Focus on the main ideas and essential details of the original text.
- Be Concise: Use clear and concise language to convey the information.

- Avoid Personal Opinions: Ensure the summary is objective and free from personal bias.
- Use Your Own Words: Paraphrase the original text to avoid plagiarism.
- Maintain Coherence: Ensure the summary is well-organized and flows logically.

OUTPUT:
Produce an enhanced version of the summary. Consider the score and reason to make it, it is important that this version is different from the previous one.
Extremely important:

- Return a plain text as output, no html
- Don't change any number, as I need them as they are in the original context.

Please don't modify the placeholders.

E.8 LLM ELO Evaluation of Insightfulness

This prompt has not been tuned using human data, as that would involve an unfair advantage to the model.

Prompt 9: ELO Insightfulness Comparison Prompt

Compare and return what insight according to the following criteria:
Insights must provide information that is:

- Insightful. The information must be interesting and relevant.
- Actionable and impactful. The information must lead to tangible follow up actions and measurable outcomes
- Data driven. The information must be concrete and derived from real data.

#####
For the output, only return the best insight (either 'Insight 1' or 'Insight 2'). No yapping
#####
All insights are related to the database:
{tables_description}
#####
Insight 1: {insight1}
#####
Insight 2: {insight2}

F Database Descriptions

For our task we used database descriptions generated with LLMs to simulate the worst case scenario. The following are the generated descriptions:

Description 1: california_schools

This database consists of three main tables: schools, satscores, and frpm. Each table contains a variety of columns with specific data formats and descriptions, providing detailed information about schools, SAT scores, and free or reduced-price meal (FRPM) eligibility.
schools Table:
This table provides comprehensive details about schools, including their identification numbers, status, location, contact information, and characteristics. Key columns include:
- CDSCode: A unique identifier for each school.
- NCESDist and NCESSchool: Identification numbers that, when combined, form a unique ID for each school according to the National Center for Educational Statistics.
- StatusType: Indicates whether a school is active, closed, merged, or pending.
- County, District, and School: Names of the county, district, and school.
- Street, City, Zip, and State: Address details of the school.
- MailStreet, MailCity, MailZip, and MailState: Mailing address details, which may default to the physical address if not provided.
- Phone and Website: Contact information for the school.
- OpenDate and ClosedDate: The dates a school opened or closed.
- Charter, CharterNum, and FundingType: Information on whether a school is a charter and its funding type.
- DOC and SOC: Codes indicating the district and school ownership types.

- EdOpsCode, EdOpsName, EILCode, and EILName: Codes and names describing the educational options and instruction levels offered.
- GSoffered and GSserved: The grade spans offered and served by the school.
- Virtual and Magnet: Indicators of whether the school offers virtual instruction or is a magnet school.
- Latitude and Longitude: Geographic coordinates of the school.
- Administrative information: Names and email addresses of administrators.
satscores Table:
This table focuses on SAT performance data for schools, including:
- cds: A reference to the CDSCode in the schools table.
- sname, dname, cname: The names of the school, district, and county.
- enroll12: Enrollment numbers for 12th grade.
- NumTstTakr: The number of students who took the SAT.
- AvgScrRead, AvgScrMath, AvgScrWrite: Average scores in reading, math, and writing.
- NumGE1500: The number of students scoring 1500 or above on the SAT.
frpm Table:
This table provides data on free or reduced-price meal eligibility, a key indicator of student poverty levels, including:
- CDSCode: A unique identifier for each school, linking back to the schools table.
- Academic Year, County Code, District Code, School Code: Identifiers and the academic year of the data.
- County Name, District Name, School Name: Names of the county, district, and school.
- District Type, School Type, Educational Option Type: Descriptions of the school and district types.
- NSLP Provision Status, Charter School (Y/N), Charter School Number, Charter Funding Type: Information on meal program status and charter school details.
- Enrollment (K-12) and Enrollment (Ages 5-17): Enrollment numbers.
- Free Meal Count and FRPM Count: Numbers of students eligible for free meals or reduced-price meals, for both K-12 and ages 5-17 categories.
Each table serves a specific purpose, from providing detailed profiles of schools to reporting on academic performance and socioeconomic indicators. Together, these tables offer a comprehensive view of the educational landscape, including school characteristics, student performance on standardized tests, and measures of economic disadvantage.

Description 2: codebase_community

This database consists of multiple tables that store various types of data related to posts, users, comments, votes, tags, and badges within a community or forum-like platform. Each table is structured with columns that have specific data formats and descriptions, providing a comprehensive overview of the interactions and content within the platform.
1. **postLinks**: This table keeps track of links between posts. Each entry has a unique post link ID, creation date, post ID, related post ID, and link type ID. The link type ID specifies the nature of the link between the posts.
2. **postHistory**: This table records the history of edits or changes made to posts. It includes information such as the post history ID, post history type ID, post ID, revision globally unique ID (GUID), creation date of the edit, user ID of the editor, detailed content of the post after the edit, comments on the edit, and the editor's display name.
3. **badges**: This table lists the badges awarded to users. Each badge has a unique ID, and the table records the user ID of the recipient, the badge name, and the date it was awarded.
4. **posts**: This table contains detailed information about each post, including post ID, post type ID, accepted answer ID, creation date, score, view count, body text, owner user ID, last activity date, ti-

tle, tags, answer count, comment count, favorite count, last editor user ID, last edit date, community owned date, parent post ID, closed date, owner display name, and last editor display name. The table provides a comprehensive overview of the content and interaction metrics for each post.
5. **users**: This table stores information about users, including user ID, reputation, account creation date, display name, last access date, website URL, location, self-introduction, views, upvotes, downvotes, account ID, age, and profile image URL. This table helps in understanding user engagement and influence within the platform.
6. **tags**: This table details the tags used within the platform, including tag ID, tag name, count of posts containing the tag, excerpt post ID, and wiki post ID. The count indicates the popularity of each tag.
7. **votes**: This table records votes cast by users on posts. It includes vote ID, post ID, vote type ID, creation date of the vote, user ID of the voter, and bounty amount (if any). This table is crucial for understanding user preferences and the quality of content.
8. **comments**: This table captures comments made on posts. It includes comment ID, post ID, score, detailed content of the comment, creation date, and user ID and display name of the commenter. The score provides insight into the sentiment or quality of the comment.
Each table is interconnected through various IDs, allowing for a relational structure that supports complex queries and analysis. This database structure enables the platform to store, retrieve, and analyze data related to user interactions, content quality, and community engagement effectively.

Description 3: debit_card_specializing

This database consists of several tables that store information related to customers, gas stations, products, consumption patterns over time, and transactions. Each table is structured with columns that have specific data formats and descriptions. Here's a breakdown of each table and the data it contains:
1. **Customers Table ('customers'):** - **CustomerID:** This is an integer field that uniquely identifies each customer. - **Segment (client segment):** A text field that categorizes customers into different segments. - **Currency:** This text field specifies the currency used by the customer.
2. **Gas Stations Table ('gasstations'):** - **GasStationID (Gas Station ID):** An integer field uniquely identifying each gas station. - **ChainID (Chain ID):** An integer field that identifies the chain to which the gas station belongs. - **Country:** A text field indicating the country where the gas station is located. - **Segment (chain segment):** A text field categorizing the gas station into different segments.
3. **Products Table ('products'):** - **ProductID (Product ID):** An integer field uniquely identifying each product. - **Description:** A text field providing a description of the product.
4. **YearMonth Table ('yearmonth'):** - **CustomerID (Customer ID):** An integer field linking the record to a specific customer. - **Date:** A text field indicating the date. - **Consumption:** A real number field representing the consumption amount.
5. **Transactions Table ('transactions_1k'):** - **TransactionID (Transaction ID):** An integer field uniquely identifying each transaction. - **Date:** A date field indicating when the transaction occurred. - **Time:** A text field specifying the time of the transaction. - **CustomerID (Customer ID):** An integer field linking the transaction to a specific customer. - **CardID (Card ID):** An integer field identifying the card used for the transaction. - **GasStationID (Gas Station ID):** An integer field linking the transaction to a specific gas station. - **ProductID (Product ID):** An integer field identifying the product involved in the transaction. - **Amount:** An integer field indicating the quantity of the product. - **Price:** A real number field indicating the price of the product. It's noted that the

total price can be calculated as the product of the Amount and Price fields.
Each table serves a specific purpose, from identifying customers and gas stations to detailing transactions and product descriptions. The structure allows for comprehensive data analysis across different dimensions such as customer behavior, product sales, and consumption patterns over time.

Description 4: european_football_2

This database consists of several tables that store detailed information about football (soccer) teams, players, matches, leagues, and countries. Each table is structured with columns that have specific data formats and descriptions, providing a comprehensive dataset for analysis or application development in the context of football.
Team Attributes ('Team_Attributes')
This table contains attributes related to football teams, including their FIFA API ID, team API ID, and various characteristics like build-up play speed, dribbling, passing, and defensive strategies. Attributes are categorized into numerical values and classes (e.g., Slow, Balanced, Fast for speed) to describe the team's playing style and strategies in different aspects of the game.
Player ('Player')
The Player table stores basic information about football players, such as their unique IDs (both API and FIFA), name, birthday, height, and weight. This table can be used to identify players and their physical attributes.
Match ('Match')
This table records details of football matches, including unique IDs for the match, country, and league, as well as the season, stage, and date of the match. It also includes the home and away team IDs, goals scored by each team, and detailed player positions and actions during the match (e.g., shots on goal, fouls committed). Betting odds from various betting agencies are also stored here, providing insights into the expected outcomes of the matches.
League ('League')
The League table contains a unique ID for each league, along with the country ID and the name of the league. This table can be used to identify different football leagues and their associated countries.
Country ('Country')
This table simply maps each country's unique ID to its name, allowing for easy identification of countries in the dataset.
Player Attributes ('Player_Attributes')
Similar to the Team Attributes table but for individual players, this table includes detailed ratings and scores for various skills and attributes (e.g., crossing, finishing, dribbling, defensive skills) as well as the player's preferred foot, attacking and defensive work rates. These attributes are rated numerically, typically on a scale from 0 to 100, providing a quantitative measure of a player's abilities and potential.
Team ('Team')
The Team table stores information about football teams, including their unique IDs (both API and FIFA), as well as their long and short names. This table can be used to identify teams and their official names.
Each table is designed to provide a detailed dataset that can be used for statistical analysis, game prediction, player evaluation, and other applications related to football. The structure and organization of the tables allow for complex queries and analyses, making it a valuable resource for researchers, analysts, and enthusiasts interested in the beautiful game.

Description 5: student_club

This database is designed to manage and track the financial, operational, and membership activities of an organization, likely a club or a small non-profit entity. It integrates several key functions related to income, expenses, budget management, events, membership, and geographic data. By centralizing this information, the database facilitates effective financial planning, monitoring of member involvement, and coordination of events.
Income Table

Description: Records the details of income received by the organization, including the source, amount, and associated member.
Columns:
Income ID (income_id): A unique identifier for each record of income (Text). Date Received (date_received): The date the funds were received (Text). Amount (amount): The amount of funds received (Integer, unit: dollar). Source (source): Indicates where the funds came from, such as dues or university allocation (Text). Notes (notes): Free-text field for additional details about the income (Text). Link to Member (link_to_member): Links to the member associated with the income (Text).
Budget Table
Description: Contains budget information for various categories and events, tracking amounts budgeted, spent, and remaining.
Columns:
Budget ID (budget_id): A unique identifier for each budget entry (Text). Category (category): The budgeted category, e.g., advertisement, food, parking (Text). Spent (spent): The total amount spent in the budgeted category (Real, unit: dollar). Remaining (remaining): Calculated as the amount budgeted minus the amount spent (Real, unit: dollar). Amount (amount): The total budgeted amount for the specified category and event (Integer, unit: dollar). Event Status (event_status): The current status of the event (Closed/Open/Planning) (Text). Link to Event (link_to_event): Links to the associated event (Text).
Zip_Code Table
Description: Stores ZIP code information, including the associated city, county, and state.
Columns:
Zip Code (zip_code): The five-digit ZIP code (Integer). Type (type): The type of ZIP code (Standard/PO Box/Unique) (Text). City (city): The city associated with the ZIP code (Text). County (county): The county associated with the ZIP code (Text). State (state): The state associated with the ZIP code (Text). Short State (short_state): The abbreviated state name (Text).

Expense Table
Description: Details expenses incurred by the organization, including amounts, descriptions, and approval status.
Columns:
Expense ID (expense_id): A unique identifier for each expense record (Text). Expense Description (expense_description): A description of what the money was spent on (Text). Expense Date (expense_date): The date the expense was incurred (Text, format: YYYY-MM-DD). Cost (cost): The dollar amount of the expense (Real, unit: dollar). Approved (approved): Indicates whether the expense was approved (True/False) (Text). Link to Member (link_to_member): Links to the member associated with the expense (Text). Link to Budget (link_to_budget): Links to the related budget entry (Text).
Member Table
Description: Contains information about members, including personal details, position, and contact information.
Columns:
Member ID (member_id): A unique identifier for each member (Text). First Name (first_name): The member's first name (Text). Last Name (last_name): The member's last name (Text). Email (email): The member's email address (Text). Position (position): The position held by the member in the organization (Text). T-Shirt Size (t_shirt_size): The member's preferred t-shirt size (Text). Phone (phone): The member's contact phone number (Text). Zip (zip): The ZIP code of the member's hometown (Integer). Link to Major (link_to_major): Links to the member's major (Text).
Attendance Table
Description: Tracks attendance of members at various events.
Columns:
Link to Event (link_to_event): Links to the event attended (Text). Link to Member (link_to_member): Links to the member who attended the event (Text).
Event Table
Description: Contains details about events organized by the organization, including

dates, types, and locations.
Columns:
Event ID (event_id): A unique identifier for each event (Text). Event Name (event_name): The name of the event (Text). Event Date (event_date): The date the event took place or is scheduled to take place (Text, format: YYYY-MM-DD). Type (type): The type of event (e.g., game, social, election) (Text). Notes (notes): Additional notes about the event (Text). Location (location): The address or name of the event location (Text). Status (status): The current status of the event (Open/Closed/Planning) (Text).
Major Table
Description: Stores information about academic majors, including the associated department and college.
Columns:
Major ID (major_id): A unique identifier for each major (Text). Major Name (major_name): The name of the major (Text). Department (department): The name of the department offering the major (Text). College (college): The name of the college housing the department (Text).

G Cost Estimation

Using GPT4o, the SQLAgent has a mean price of $0.072 for each query. The Hypothesis Generator for our method has a mean price of $0.113 per use.

On average, each insight generated by our method has 7.25 subquestions, resulting in a mean cost of $0.637 per insight.

This is a great improvement over the previous manual method of generating insights, which amounts to an approximate of $140 per insight (as estimated in our company).

Additionally, the cost of GPT-4o has recently dropped by 79% in a few months. We therefore expect the cost of LLM to continue to drop, given also recent models (e.g., Deepseek) that can reach top performance at a much lower cost — 10x to 30x lower compared to GPT-4o.

H Use of Coding Assistant

ChatGPT was used as a coding assistant for the elaboration of the results and the experiments.