applied sciences

MDPI

Article

RAG-Grounded Natural-Language-to-SQL for On-Premise Deployment in Regulated Batch Manufacturing

Luis Rabelo [1], Juan Zamora-Aguas [1], Fahad Alasim [2,*], Jose Vazquez [3], Milena Urbina [1], Gustavo Manfredini [1], Sean Doolittle [1] and Edgar Gutierrez-Franco [4]

[1] Department of Industrial Engineering and Management Systems, University of Central Florida, Orlando, FL 32816, USA; luis.rabelo@ucf.edu (L.R.); juanpablo.zamoraaguas@ucf.edu (J.Z.-A.); ubrinam@gmail.com (M.U.); gustavo.manfredini@ucf.edu (G.M.); sean.doolittle@ucf.edu (S.D.)
[2] Department of Industrial Engineering, King Saud University, Riyadh 11421, Saudi Arabia
[3] Altra Systems, Orlando, FL 32827, USA; j.vazquez@altrasystems.com
[4] Center for Transportation and Logistics, Massachusetts Institute of Technology, Cambridge, MA 02142, USA; edfranco@mit.edu
* Correspondence: falasim@ksu.edu.sa

Abstract

Natural-language access to structured manufacturing data could let plant personnel query batch records without structured query language (SQL) expertise, yet realistic multi-table industrial schemas remain hard for text-to-SQL systems, and regulated environments demand data sovereignty, traceability, and validatable behavior that cloud-hosted models cannot satisfy. This study presents a locally deployable large language model (LLM) system that translates questions into SQL over an ISA-88-aligned database built from a pharmaceutical tablet dataset of 1005 batches. The pipeline combines schema-aware prompting, retrieval-augmented generation (RAG) over validated query examples, execution-based checking, confidence scoring, and human review. Evaluation on 48 questions across three complexity tiers compared a schema-only baseline with five enhancements: controlled vocabularies, retrieved examples, analytical views, execution-guided self-consistency, and schema-specialized fine-tuning. Retrieved examples roughly doubled execution accuracy for two open-weight models, from 23–25% to 46–48%, without reducing accuracy on any question; gains concentrated the 15 two- and three-table join questions, reaching 80%. Fine-tuning a seven-billion-parameter code model on execution-validated synthetic queries raised its accuracy from 19% to 44%, and, combined with retrieval, it achieved the highest observed accuracy on the hardest tier. Results show that a lightweight, auditable, locally deployable pipeline can support regulated manufacturing analytics while flagging outputs that require human validation.

Keywords: text-to-SQL; retrieval-augmented generation; local large language models; ISA-88; batch manufacturing; data sovereignty

Check for updates

Academic Editor: Dong Hyun Jeong

Received: 29 September 2026
Revised: 30 October 2026
Accepted: 3 November 2026
Published: 5 November 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

Pharmaceutical and other regulated manufacturers increasingly depend on data-driven decision-making [1,2], yet extracting timely insight from the large, complex process data generated on the manufacturing floor remains difficult. A single tablet-compression line, for instance, produces high-frequency sensor time series alongside raw-material attributes, in-process controls, and laboratory quality results for every batch [3]. Traditional analytics workflows require specialists to translate operational questions into SQL queries that span

many interdependent tables, debug their syntax, and validate the results manually. Natural-language interfaces to databases have long been proposed to remove this barrier [4]. Recent shop-floor studies confirm that limited digital literacy and rigid interfaces still prevent operators from querying manufacturing data without external technical support [5]. This manual path can take hours or days, creates a dependency on a small pool of SQL-proficient staff, and delays the interventions that prevent costly batch failures and compliance issues.

Large language models (LLMs) could shorten this path. They are already being adapted to manufacturing tasks, from domain-specific fine-tuning [6] to conversational machine monitoring over real-time shop-floor data [5]. Cloud-hosted natural-language and LLM services, however, conflict with the data-sovereignty, real-time, and compliance requirements of regulated production. Sending prompts that contain proprietary process records to third-party providers exposes them to confidentiality and data-leakage risks that are well documented in the LLM security literature [7]. Standards such as ISA-88 for batch control emphasize structured data, traceable process flows, and validated reporting [8,9]. Generative models, by contrast, introduce the risk of hallucinations: outputs that are plausible but factually incorrect or unfaithful to their source [10,11]. For organizations operating under good manufacturing practice, sending batch data to external providers is frequently not permissible. Any model output that informs a regulated decision must also be traceable and verifiable, consistent with the ALCOA+ data-integrity principles and computerized-system requirements enforced by regulators [12–14]. Regulators have recently made this position explicit for generative AI. The draft EU GMP Annex 22 excludes LLMs from critical GMP applications and confines them to non-critical uses in which qualified personnel remain responsible for the suitability of the output [15].

These constraints motivate a locally deployed alternative. A schema-aware LLM can convert natural-language questions into SQL within client-controlled infrastructure if it is grounded in validated documentation through retrieval-augmented generation (RAG) [16] and constrained by a human-in-the-loop review step, in line with the human-centric orientation of Industry 5.0 [17]. We are careful not to overstate what such a system can achieve. On realistic, multi-table schemas, text-to-SQL is far from solved [18,19]. On the enterprise-grade Spider 2.0 benchmark, an agentic framework built on a frontier reasoning model solved only 21.3% of tasks, against 91.2% on the earlier Spider 1.0 [20], and locally hostable models trail cloud frontier systems on the hardest queries. The contribution of this work is therefore not state-of-the-art accuracy. It is a demonstration that a deliberately lightweight, auditable pipeline is feasible, usable, and reliable enough to be valuable in a setting where the realistic alternative is no AI assistance at all.

Our contributions are fourfold. First, the approach prioritizes local deployment and data sovereignty: the architecture is designed so that the LLM, exemplar store, and database all run within client-controlled infrastructure, removing cloud dependencies and keeping sensitive manufacturing data inside the organization's controlled environment, and all evaluated models are open-weight and locally deployable. Second, it employs a schema-aware, RAG-grounded pipeline that retrieves validated query examples from a local exemplar store before generation, improving grounding for complex cross-table questions. Third, it incorporates reliability safeguards: execution checking of every generated query, confidence scoring from the agreement of executed candidates, and a human-in-the-loop review step, complemented in the offline evaluation by execution-based fidelity measurement against ground-truth results. Together, these keep outputs advisory and auditable rather than prescriptive, as current regulatory guidance expects for generative models [15]. Finally, it provides an empirical evaluation under realistic constraints. A tiered 48-question benchmark over the ISA-88 schema compares three context configurations (schema-only, controlled vocabularies, and retrieved validated examples) across two locally hostable open-weight

models. The evaluation also covers execution-guided self-consistency, an adaptation of self-consistency decoding [21] to SQL generation, and a schema-specialized fine-tuned model, establishing which interventions improve accuracy and which do not.

The remainder of the paper is organized as follows. Section 2 reviews related work across four dimensions and identifies the gap addressed here. Section 3 describes the system design and implementation. Section 4 defines the evaluation methodology and Section 5 reports the results. Section 6 discusses the findings against the literature, and Sections 7 and 8 present limitations, future work, and conclusions.

2. Background and Related Work

Large language models have opened new opportunities for data access, analysis, and decision support in industrial environments. In regulated batch manufacturing, where structured data, compliance, and data integrity are critical, LLMs offer a way to bridge natural-language questions and complex operational datasets. This section reviews LLM-based interfaces, hallucination mitigation, retrieval-augmented generation, and SQL generation, and situates these advances within ISA-88 structured manufacturing systems.

2.1. Key Dimensions of LLM-Based Industrial and Pharmaceutical Analytics

2.1.1. LLMs as Natural-Language Interfaces

Recent work positions LLMs as interaction layers that translate unstructured user intent into structured analytical operations rather than as replacements for existing pipelines. In pharmaceutical settings, Salami et al. [22] show that foundational models can support deviation investigations by extracting root causes and comparing similar events through semantic embeddings, reducing analyst effort while leaving some reliability concerns. Beyond deviations, Kadi et al. [23] combine LLMs with RAG to extract IDMP-aligned fields from SmPC documents at high F1, and Kim et al. [24] answer questions over FDA and ICH guidance with a dual-track retrieval framework. Vieira-Vieira et al. [25] introduce PRINCE, a multi-agent engine that unifies preclinical safety data through LLMs, RAG, and text-to-SQL, while Taskiran et al. [26] integrate ontologies with LLM reasoning in a drug-discovery chatbot. Industrial neighbors extend the same idea to the plant floor: an LLM-enabled agent architecture provides real-time decision support over batch-process sensor streams in a clean-in-place case study [27], and a knowledge-graph-grounded interface lets manufacturing planners query simulation output in natural language while preserving human oversight [28]. Collectively, these studies show a clear trend toward LLMs as conversational front-ends, but most target unstructured text, sensor streams, or preclinical data rather than structured relational batch-manufacturing databases. Table 1 summarizes the main contributions.

Table 1. Main contributions: LLMs as natural-language interfaces for industrial and pharmaceutical analytics.

Article	Main Approach	Tools/Techniques	Core Contribution
Wan et al. [29]	Domain-centric Q&A via hybrid RAG	Knowledge graphs, vector embeddings, hybrid RAG, semantic constraints, prompt engineering	LLMs as effective NL interfaces for smart-manufacturing analytics when grounded with domain knowledge
Salami et al. [22]	Generative extraction + similarity search	GPT-family LLMs, embedding models, cosine similarity, OCR preprocessing	NL access to deviation reports for investigation support and knowledge reuse in pharma manufacturing

Table 1. *Cont.*

Article	Main Approach	Tools/Techniques	Core Contribution
Oyekan et al. [30]	Decision-guidance framework	Ontologies, knowledge graphs, LLM prompting, inference-speed/accuracy metrics	Positions LLMs as NL interfaces for human–robot collaboration and industrial reasoning
Wang et al. [31]	Multimodal LLM framework	Multimodal transformers, dynamic semantic tokenization, cross-modal alignment	Extends NL interfaces to multimodal factory data for perception and decision-making
González-Potes et al. [27]	Agent-based real-time decision support	LLM analytics agent, RAG, deterministic agents, human operators	NL access over batch-process sensor streams (clean-in-place); not text-to-SQL over relational records
Thogaru et al. [28]	KG-grounded NL interface	Knowledge graphs, LLM agents, iterative reasoning	NL querying of manufacturing simulation output with human oversight
Mahmud et al. [32]	Rule-based NL query processing	Rule-based NLP parsing, handcrafted grammar, SQL generation	Early baseline for NL access to industrial databases without LLMs

2.1.2. Hallucinations, RAG, and Hybrid Architectures

A dominant concern in regulated applications is the risk of hallucinated or unverifiable output. The literature responds by embedding LLMs in retrieval-augmented and hybrid architectures that ground generation in external knowledge. Salami et al. [22] analyze how hallucinations can mislead deviation investigations when models infer missing detail, and Kadi et al. [23] motivate RAG and careful prompting to constrain extraction. Fan et al. [33] survey retrieval-augmented LLMs, showing that retrieval and augmentation design strongly affect reliability and that high-quality, well-aligned evidence substantially reduces hallucination. Domain-specific work extends this: ClinicalRAG cross-checks labeling statements against clinical tables through hierarchical RAG [34]; QA-RAG retrieves with both the user query and a generated hypothetical answer [24]; and Wan et al. [29] combine knowledge graphs with dense retrieval for manufacturing Q&A, while Choi and Jeong [35] propose frameworks for keeping RAG knowledge current. These works, however, target document-level Q&A and summarization rather than executable SQL over structured production data. Table 2 summarizes the contributions.

Table 2. Main contributions: hallucinations, retrieval-augmented generation, and hybrid architectures.

Article	Main Approach	Tools/Techniques	Core Contribution
Wan et al. [29]	Hybrid KG-vector RAG	Knowledge graphs, vector search, hybrid retrieval orchestration, constraint-based alignment	Mitigates hallucinations and context loss by combining symbolic reasoning with scalable retrieval
Fan et al. [33]	RA-LLM survey	Sparse/dense retrieval, RAG architectures, retrieval-generation coupling	Establishes RAG as a core strategy to reduce hallucination and stale knowledge
Kim et al. [24]	QA-RAG (dual-track retrieval)	Dual-track retrieval, RAG pipelines, generative answer reformulation	Improves robustness in regulatory tasks by retrieving with queries and generated answers

Table 2. *Cont.*

Article	Main Approach	Tools/Techniques	Core Contribution
Zhou et al. [34]	Hierarchical RAG (ClinicalRAG)	Hierarchical retrieval, statement decomposition, table-based verification, self-consistency	Controls hallucinations via statement-level verification against structured evidence
Salami et al. [22]	Empirical LLM evaluation	Prompt-based extraction, embedding similarity, manual ground-truth validation	Demonstrates the reasoning-hallucination trade-off in real manufacturing data

2.1.3. Text-to-SQL, ISA-88, and Structured Batch Data

Text-to-SQL is the natural bridge between natural-language interfaces and structured manufacturing databases. Surveys trace the shift from rule-based parsers to neural and LLM-based approaches that use schema-aware prompting and RAG [19,36,37], identifying schema linking, metadata retrieval, and hallucination control as critical open challenges in complex, domain-limited deployments. Early rule-based interfaces [32] demonstrated feasibility but lacked flexibility, and large benchmarks such as BIRD [18] expose how hard realistic SQL grounding is, with even strong models achieving only modest execution accuracy on noisy, multi-table databases owing to incorrect joins, value-type errors, and missing domain knowledge.

It is important to situate the present work against the current frontier, which has moved well beyond single-shot prompting. On the enterprise-grade Spider 2.0 benchmark, accuracy collapses relative to the older Spider 1.0, and leading systems are multi-call agentic pipelines: ReFoRCE [38] tops the leaderboard through systematic schema exploration and self-correction, Alpha-SQL [39] uses Monte Carlo Tree Search for zero-shot generation, and autonomous schema-linking methods address very large schemas. These systems optimize benchmark accuracy at the cost of many model and database calls per question, non-determinism, and limited auditability. In parallel, small, locally hostable models have become viable: a 7B code-specialized model with lightweight query pre-synthesis [40] reaches competitive execution accuracy, although self-hosted models still trail cloud frontier systems on the hardest queries. Recent analyses also caution that text-to-SQL benchmark annotations themselves contain substantial errors, which affects how leaderboard numbers should be interpreted. The design adopted here deliberately trades agentic accuracy for determinism and auditability, and uses a code-specialized small model as a comparator to reflect current local-model practice.

Manufacturing data add further complexity. Virta et al. [9] describe how ISA-88 and ISA-95 models are implemented through OPC UA and service-oriented architectures, emphasizing that batch records are deeply hierarchical and semantically interdependent across unit procedures, phases, and equipment modules; Murali and Anu Priya [41] show that pharmaceutical process data exhibit multivariate, nonlinear and time-varying behavior. Vichev and Marchev [42] demonstrate that embedding models differ widely in retrieving correct schema components and that retrieval errors cascade into malformed SQL, reinforcing the need for schema-aware prompting and dynamic retrieval. PRINCE [25] shows that SQL-generating pipelines can operate in regulated R&D contexts, but on preclinical rather than ISA-88 batch data. No prior LLM-based system has targeted ISA-88-structured batch-manufacturing records, and existing text-to-SQL work relies on generic benchmarks that do not reflect regulated batch environments. Table 3 summarizes the contributions.

Table 3. Main contributions: text-to-SQL, ISA-88, and structured batch manufacturing data.

Article	Main Approach	Tools/Techniques	Core Contribution
Virta et al. [9]	Standards-based integration	OPC UA, SOA middleware, ISA-88/95 models, MES-PCS integration	Interoperable structured batch-data architectures enabling analytics and AI access
Vichev and Marchev [42]	RAG-enhanced text-to-SQL	Embedding models, sentence-transformer fine-tuning, RAG context selection	Retrieval-based context augmentation improves NL-to-SQL accuracy
Hong et al. [19]	Text-to-SQL survey	LLM-based SQL generation, schema linking, benchmark datasets	Synthesizes architectures, datasets, and evaluation for NL database interfaces
Mohammadjafari et al. [36]	Review of LLM text-to-SQL	Schema-aware prompting, retrieval-augmented prompting, schema linking	Identifies schema linking and hallucination control as open challenges
Deng et al. [38]	Agentic text-to-SQL (Spider 2.0 SOTA)	Schema exploration, self-correction, voting/refinement	Frontier accuracy via heavyweight multi-call pipelines; low auditability
Yan et al. [40]	Small-model text-to-SQL	Query pre-synthesis, few-shot prompting, code-specialized 7B model	Competitive accuracy from locally hostable small models
Kumar et al. [37]	Deep-learning text-to-SQL survey	CNN/RNN/transformer models, benchmark datasets, exact-match metrics	Historical baselines and dataset limitations for NLIDB systems

2.1.4. AI in Regulated Environments

Regulated manufacturing stresses industrial LLM systems with stringent demands for traceability, accuracy, and compliance, and the literature shows a shift from generative convenience toward verification-centric, compliance-aware architectures. Zhou et al. [34] demonstrate safe label quality control through hierarchical RAG with explicit evidence checking; Kim et al. [24] address regulatory change through enhanced retrieval; and Kadi et al. [23] convert SmPC documents into IDMP-compliant data with expert-in-the-loop validation. From an architectural standpoint, Dogruluk and Acikgoz [43] survey edge-centric federated learning for industrial LLMs, arguing that privacy-preserving, low-latency intelligence should run at the industrial edge and noting compliance with standards such as IEC 62443 [44]. Kavasidis et al. [14] predict ALCOA+ data-integrity compliance from heterogeneous plant data, reinforcing that pharma AI must respect data-integrity principles and remain auditable. The local system developed here aligns with these trends through an architecture designed to run an open-weight LLM that performs NL-to-SQL translation inside the plant network. Table 4 summarizes the contributions.

Table 4. Main contributions: AI in regulated environments.

Article	Main Approach	Tools/Techniques	Core Contribution
Zhou et al. [34]	Verification-centric RAG	Hierarchical RAG, table QA, UI-supported human review	Safe LLM use for pharmaceutical label QC with explicit evidence checking
Kim et al. [24]	QA-RAG for compliance	RAG pipelines, generative reformulation, compliance-oriented retrieval	Addresses regulatory change and traceability via enhanced retrieval

Table 4. *Cont.*

Article	Main Approach	Tools/Techniques	Core Contribution
Kadi et al. [23]	Structured extraction + expert review	LLM extraction, IDMP schema mapping, expert-in-the-loop validation	Converts SmPC documents into IDMP-compliant data with human validation
Kavasidis et al. [14]	Compliance-prediction modeling	Transformer architectures, compliance-scoring models	Predicts ALCOA+ data-integrity compliance using deep learning
Salami et al. [22]	Risk-aware LLM deployment	Prompting strategies, embedding similarity, human-validation workflows	Highlights governance and hallucination risks in regulated manufacturing

2.2. Summary and Identification of the Gap

Three research streams converge on this problem but none addresses it directly. First, text-to-SQL has advanced rapidly, yet recent benchmarks show that the task is far from solved on realistic schemas: leading systems reach only modest accuracy on enterprise-grade Spider 2.0, and the gains come from heavyweight, multi-call agentic pipelines optimized for benchmark accuracy rather than for auditability or local deployment. Second, LLM-based natural-language interfaces for manufacturing are now an active area, including agent-based decision support for batch processes and knowledge-graph-grounded interfaces for production planning, but these target document Q&A, sensor and SCADA streams, or simulation output, not text-to-SQL over the hierarchical relational records defined by ISA-88. Third, privacy-preserving and on-premise LLM deployment is increasingly studied for industrial settings, yet existing demonstrations are not evaluated against the validation, traceability, and usability requirements of regulated manufacturing. No prior work, to our knowledge, combines all three: a locally deployed, schema-aware, RAG-grounded text-to-SQL system over an ISA-88-structured batch database, evaluated for SQL-level fidelity under regulated-environment constraints and designed for use by non-programmer plant personnel.

3. System Design and Implementation

This section describes the architecture and approach used to design and implement a localized AI-driven analytical system for pharmaceutical manufacturing and quality contexts. It explains how the components translate natural-language questions into executable SQL over an ISA-88-aligned batch database, in a form suitable for regulated manufacturing. The architecture is designed so that all components run on a local server under client control, preserving data sovereignty.

The architecture is intentionally lightweight rather than maximally accurate. State-of-the-art text-to-SQL systems achieve their results through multi-call agentic pipelines that iteratively explore the schema, execute candidate queries, and self-correct, often issuing many model and database calls per question. In a validated, regulated environment, these properties are liabilities as much as assets: each additional autonomous step expands the surface that must be qualified, monitored for drift, and explained to an auditor, and non-deterministic search complicates reproducibility and change control. We therefore adopt a deterministic, single-pass generation step grounded by retrieval and bounded by an explicit human-in-the-loop review step, prioritizing predictability, traceability, and ease of validation over benchmark accuracy; a multi-sample, execution-guided variant is evaluated separately as a controlled refinement (Section 4.5). For the same reason of deployability under client control, all models are open-weight and can be served on-premise in 4-bit

quantization on a single workstation-class GPU, and a general-purpose model is compared with a code-specialized model of the same architecture, alongside a smaller model fine-tuned on the schema (Section 3.5). The aim is not to surpass cloud frontier systems on accuracy, which would be unrealistic given the gap reported between frontier and locally deployable models on complex queries, but to characterize the accuracy and reliability that an auditable pipeline built on locally deployable models can deliver.

3.1. System Overview

The system integrates natural-language processing, SQL generation. and retrieval to improve data accessibility and accuracy while remaining compatible with ISA-88 batch-control practice. Altra Systems' experience in ISA-88 standards, MES integration. and SQL-based data warehousing informed the selection of components so that AI-generated outputs remain consistent with established batch records and equipment models. As shown in Figure 1, the system comprises four main components. The LLM engine is a locally served open-weight model that receives user questions and, using schema-aware prompts, generates candidate SQL.

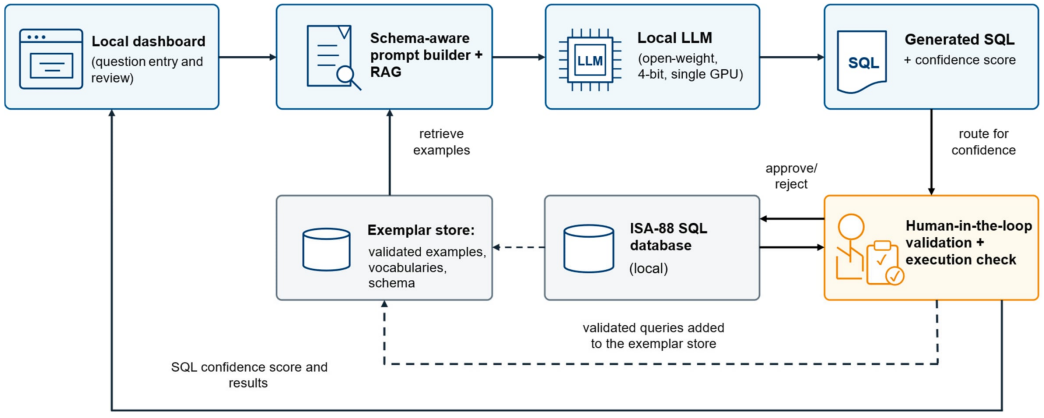


Figure 1. System architecture for the local LLM-driven manufacturing analytics tool, showing the flow from the user interface through the LLM and retrieval components to the locally hosted ISA-88 SQL database and exemplar store. Light-blue boxes indicate interface and query-generation components, gray boxes indicate local data resources, and the orange box indicates human-in-the-loop validation and execution checking. Solid arrows show the main query and result flow; dashed arrows show information supplied to and updates of the exemplar store, including validated queries returned after review.

The retrieval layer augments the prompt with controlled domain vocabularies and a small number of validated worked examples, selected per question by lexical similarity from a curated store held out from the benchmark; this improves grounding and reduces hallucinations on complex questions. The SQL database layer is a locally hosted relational database containing the ISA-88-aligned dataset, against which generated queries are executed and benchmarked. Finally, the user interface is designed as a lightweight dashboard where users submit questions and inspect the generated SQL, the returned result with a plain-language summary. and, when self-consistency is enabled, the confidence score.

The interface sits at the top, the LLM and retrieval components in the middle logic layer, and the relational database and exemplar store at the bottom data layer. This separation supports modular tuning, such as improving prompts or retrieval without changing the interface, and keeps control over where regulated data resides.

While Figure 1 sets out the static component architecture, Figure 2 traces how these components act on a single query.

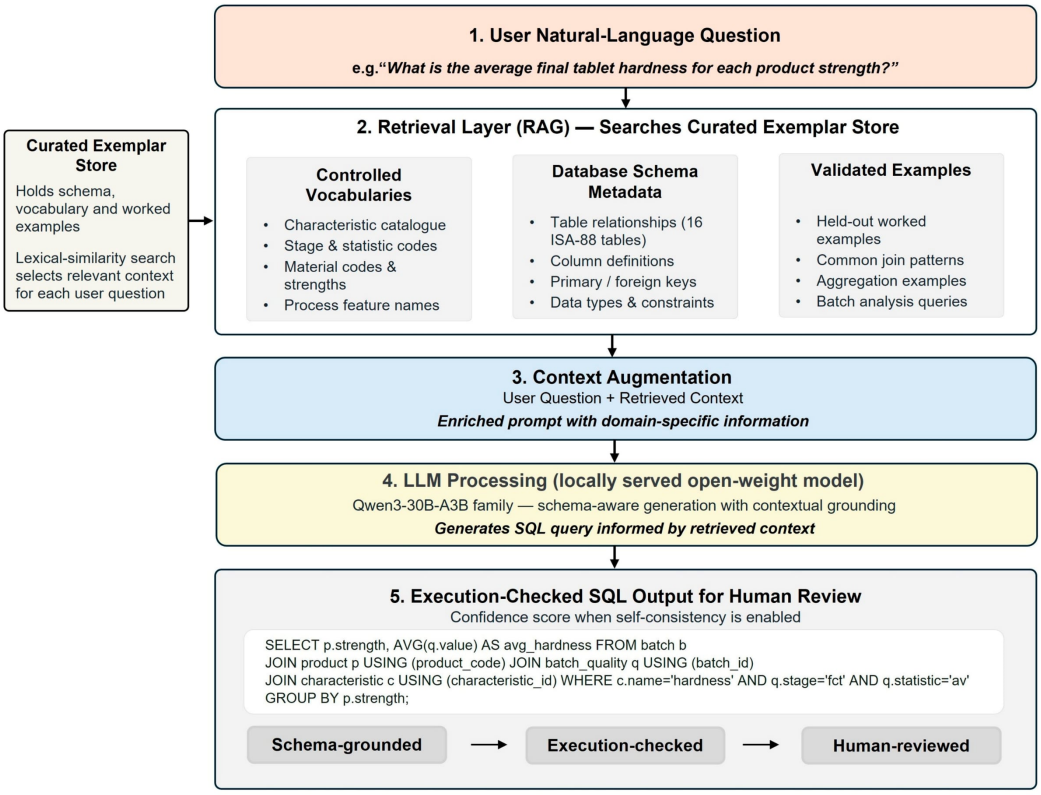


Figure 2. RAG-grounded pipeline for the local LLM-driven manufacturing analytics tool. A user's natural-language question (1) triggers retrieval (2) from a curated store, held out from the benchmark, of schema metadata, controlled domain vocabularies, and validated worked examples for the 16-table ISA-88-aligned pharmaceutical tablet database. The retrieved context augments the schema-aware prompt (3); a locally served open-weight model (4) then generates SQL, which is execution-checked and returned for human review, with a confidence score when self-consistency is enabled (5).

A user's natural-language question is first grounded by retrieval from the exemplar store; the retrieved schema metadata, controlled vocabularies. and validated query patterns augment the prompt, and a locally served open-weight model then generates SQL that is execution-checked and returned for human review, with a confidence score when self-consistency is enabled. The diagram thus shows how the elements described above combine into a single deterministic, retrieval-grounded pass from question to validated output.

3.2. Dataset

To evaluate the system on pharmaceutical data, we use the openly published tablet-manufacturing dataset of Žagar and Mihelič [3], comprising 1005 production batches of a film-coated tablet collected over roughly thirty months, with incoming raw-material analyses, intermediate and final product quality, and tablet-compression process time series at ten-second resolution. Because the published data are delivered as flat files oriented to quality prediction, we transformed them into a normalized, ISA-88-aligned schema of sixteen interdependent tables spanning product and recipe, equipment hierarchy and sensors, material genealogy, procedural execution (a compression operation with start-up and production phases), process data. and quality results; Appendix B (Figure A1) gives the full schema, organized into the ISA-88 sub-models with the batch table as the central hub.

3.3. Prompt Engineering and Query Pipeline

The pipeline converts a user question into executable SQL through controlled prompt steps. The model is first primed with schema information (table names, relationships, primary keys, and key attributes) so it can reason about which tables and joins are relevant. Structured prompt templates then guide it to identify the required tables, construct a syntactically valid query, and avoid inventing columns or tables. Retrieval is treated as an experimental condition: in the retrieval configuration, the system adds controlled domain vocabularies to the schema context and retrieves validated worked examples from the exemplar store to clarify ambiguous terms (for example, batch outcome versus equipment failure); in the schema-only configuration, this retrieval is disabled. In the implementation evaluated in this study, exemplars are ranked by lexical similarity (token overlap weighted by inverse document frequency; Appendix A.3), which keeps retrieval deterministic and free of additional model dependencies. After generation, the query is executed against the database and presented with its result for human review; execution checking verifies that the generated query runs without error. Execution checking and the confidence score of the self-consistency procedure (Section 4.4) apply to previously unseen questions, while comparison with analyst-validated reference SQL is used in the offline evaluation (Section 4.3). Low-confidence or divergent outputs are intended to be flagged in the interface for review or rejection, keeping humans in the loop for critical decisions.

3.4. User Interface

The interface is designed as a simple dashboard for engineers, quality specialists, and managers who may not have programming backgrounds. Users enter a natural-language question and the dashboard is designed to display side-by-side the generated SQL, the returned result with a short plain-language summary, and, when self-consistency is enabled, the confidence score of the query. When retrieval is active, it also lists the worked examples and vocabulary entries supplied to the prompt, increasing transparency. Review is designed to take place in the same dashboard: users review each generated query and its result and mark the output as correct, partially correct, or incorrect, and queries marked correct are intended to be added to the exemplar store for subsequent prompt tuning and retrieval adjustment.

3.5. Models Evaluated

To reflect current local-model practice, the system is evaluated with open-weight models that can be served entirely on-premise. The primary comparison uses two members of the Qwen3 30B-A3B family [45]: a general-purpose instruction model (Qwen3-30B-A3B-Instruct) and a code-specialized model of the same architecture (Qwen3-Coder-30B-A3B-Instruct). Both are mixture-of-experts models with 30.5 billion total but only about 3.3 billion active parameters per token, so that in 4-bit quantization, they run on a single workstation-class GPU; sharing architecture and scale, they isolate the effect of code specialization. To probe schema adaptation, a smaller code model, Qwen2.5-Coder-7B-Instruct [46] is additionally fine-tuned on the schema (Section 4.6). All models are served under identical retrieval and prompting conditions, and serving and version details are given in Appendix A.

4. Evaluation Methodology

4.1. Datasets and Question Bank

The system is evaluated on the ISA-88-aligned relational database derived from the pharmaceutical tablet-manufacturing data of Žagar and Mihelič [3], comprising 1005 production batches across 25 product sub-families. Over the pharmaceutical schema, we con-

structed a benchmark of 48 natural-language questions reflecting realistic operational and quality queries, spanning batch genealogy, raw-material and product quality, process features, equipment and sensor relationships, campaign roll-ups, and temporal analysis. Each question is labeled with one of three complexity tiers: single-table or simple-aggregation (T1, n = 13), two- to three-table joins (T2, n = 15), and multi-join queries involving aggregation, filtering, temporal logic, or sub-queries (T3, n = 20). The full question bank, with per-question tier and category labels, is provided in Appendix B. The benchmark is deliberately domain-specific rather than drawn from generic corpora such as Spider or BIRD, because the contribution concerns querying ISA-88-structured batch records; it is extensible, and the released version provides the seed set used here.

4.2. Ground-Truth Construction

Each question is paired with a reference (gold) SQL statement authored against the schema and validated by execution: every reference query was run against the database and its returned result set inspected for correctness, and a programmatic verifier confirms that all 48 queries execute and return the intended rows. Because recent analysis has shown that text-to-SQL benchmark annotations frequently contain errors [47], this execution-based verification was treated as a required step rather than an afterthought. Where a quantity could not be reconstructed exactly from the raw time series without the original feature-extraction routines, the published per-batch values of the source dataset were taken as ground truth, and the raw series retained separately (Section 3.2).

4.3. Metrics

The primary metric is execution accuracy: a generated query is correct if, when executed, it returns the same result set as the reference query. Result sets are compared as multisets of rows, insensitive to row and column order and with floating-point values rounded to four decimal places, so that a logically equivalent query phrased differently is still scored as correct. Column order is neutralized within each row: the values of a row are sorted into a canonical order, so that each row is compared as a multiset of its values, and column names or aliases are not used in the comparison; result sets with different numbers of columns therefore do not match. As a partial-credit measure, we report fidelity, the Jaccard overlap between the returned and reference row multisets: the size of their multiset intersection divided by the size of their multiset union, with each distinct row counted as many times as it occurs, so a duplicated row contributes to the overlap only up to the number of times it occurs in both result sets. Fidelity is 1 when both result sets are empty and 0 when only one is empty; a query that fails to execute, and a missing prediction, score 0. All 48 reference result sets are non-empty. Each outcome is assigned to one category (exact match; value mismatch, where the query executes but returns different data; SQL or execution error, such as referencing a non-existent column or an invalid join; empty result; or no prediction) to support the failure-mode analysis in Section 5.

4.4. Confidence Scoring and Human Oversight

Confidence scores are produced by the execution-guided self-consistency procedure (Section 4.5) as the fraction of sampled candidates whose executed result set agrees with the selected one, with candidates that fail to execute counted in the denominator (Appendix A.5); in the single-pass configurations, execution checking provides the signal for review. Section 5.3 assesses the self-consistency scores of both models directly, without a review threshold. In the offline evaluation, execution-based comparison against ground truth provides the objective correctness signal, while the confidence mechanism is assessed by its association with that signal.

4.5. Models and Experimental Configurations

The system was evaluated with locally deployable, open-weight models so that every reported configuration could, in principle, run on-premise without a cloud dependency. For the evaluation, the two Qwen3 models were accessed through a hosted inference service serving the published open weights, and the Qwen2.5-Coder-7B models were fine-tuned and evaluated in a cloud-hosted GPU notebook; because the source data are public, the schema, questions, and database could be processed outside plant infrastructure (Appendix A.2). The primary comparison used two members of the Qwen3 30B-A3B family: a general-purpose instruction model (Qwen3-30B-A3B-Instruct), and a code-specialized model of the same architecture (Qwen3-Coder-30B-A3B-Instruct). Both are mixture-of-experts models with 30.5 billion total but only about 3.3 billion active parameters per token, so that in 4-bit quantization, they occupy on the order of 18 GB and run on a single workstation-class GPU. Using two models that share architecture, scale, and active-parameter count and differ only in training emphasis isolates the effect of code specialization.

Three context configurations were compared, holding the model, prompt template, and scoring constant and varying only the information supplied alongside the question. The schema-only configuration supplied the table, column, and foreign-key definitions. The vocabulary configuration additionally supplied controlled domain vocabularies (the characteristic catalog, the stage and statistic enumerations, the material codes, the product strengths, and the process-feature names). The retrieval configuration additionally supplied a small number of validated worked examples retrieved per question from a held-out exemplar store by a lightweight lexical-similarity retriever, drawn from a pool disjoint from the evaluation set so that no benchmark answer was leaked. Two further refinements were evaluated as additions to the retrieval configuration. First, a set of curated analytical views was added to the database (one row per batch, with the entity-attribute-value quality measurements pivoted into named columns), present in both contexts while the reference queries continued to use the base tables. Second, an execution-guided self-consistency procedure was evaluated, in which several candidate queries were sampled, executed against the database, repaired when they raised an error or returned no rows, and resolved by majority vote over executed result sets, with the fraction of candidates agreeing on the selected result reported as a confidence score. All sampling used a temperature of zero for the deterministic first candidate and a moderate temperature for additional candidates, and the database, schema context, and scoring were identical across configurations. Table 5 summarizes the models and context configurations evaluated. The fine-tuned model is described in Section 4.6.

Table 5. Locally deployable models and context configurations evaluated.

Element	Setting
General model	Qwen3-30B-A3B-Instruct (30.5B total, ~3.3B active)
Code-specialized model	Qwen3-Coder-30B-A3B-Instruct (same architecture)
Fine-tuned model (Section 4.6)	Qwen2.5-Coder-7B-Instruct, QLoRA-adapted
Untuned control (Section 4.6)	Qwen2.5-Coder-7B-Instruct, same 4-bit base without adapter
Schema-only context	table, column and foreign-key definitions
+vocabulary	+controlled domain vocabularies
+retrieval	+validated worked examples (held out from the benchmark)
Refinement: analytical views	wide reporting views, present in both contexts
Refinement: self-consistency	sample, execute, repair, majority vote, confidence score

4.6. Model Adaptation by Fine-Tuning

To address the residual failures on the most complex queries (Section 5.4), a smaller open-weight model was fine-tuned on the schema. The base model was Qwen2.5-Coder-7B-Instruct, selected because a seven-billion-parameter model is inexpensive to fine-tune on a single 16 GB GPU and is straightforward to deploy on-premise, making it the most faithful artifact for the deployment setting.

The training set was generated synthetically and validated by execution. A library of parameterized templates spanning the schema's query patterns (single-table aggregates, two- and three-table joins, raw-material genealogy, the stage- and statistic-filtered quality measurements, process-feature filters, grouped aggregation with HAVING, sub-queries comparing a group to an overall average, conditional pass-rate calculations, stage comparisons, temporal grouping, and joins over the analytical views) was instantiated over real entities drawn from the database, so that each generated query was correct by construction; every query was then executed and retained only if it ran successfully. Each template was tagged by complexity tier, and generation enforced an intentionally compositional profile (approximately 10%, 35%, and 55% for tiers T1, T2, and T3) rather than the distribution that is easiest to enumerate, because the complex T3 queries were the principal failure mode and therefore required over-representation; per-family balancing within each tier prevented high-cardinality lookup patterns from dominating. Natural-language diversity was increased by generating several meaning-preserving rewrites of each question with a language model while leaving the validated SQL unchanged, so that label correctness was retained. Generated questions whose normalized text matched one of the 48 benchmark questions or one of the retrieval exemplars were discarded, and the data were formatted using the same prompt template employed at inference. The resulting set comprised 1112 question–SQL pairs (111 T1, 389 T2, and 612 T3) derived from 180 distinct SQL queries with approximately six phrasings each. The pairs were shuffled with a fixed seed and split at random, at the level of individual pairs, into 1001 training and 111 validation pairs (10%).

The model was adapted using QLoRA [48]: the base was loaded in 4-bit precision, and low-rank adapters of rank 16 were trained on the attention and feed-forward projection matrices, with the loss computed only on the SQL completion. Training ran for two epochs with an effective batch size of eight; the validation loss decreased monotonically and plateaued below 0.01, indicating stable convergence on the synthetic distribution. The resulting adapter is a few tens of megabytes and is loaded on top of the open base model, so the adapted system remains fully self-hostable. To isolate the effect of adaptation, the original Qwen2.5-Coder-7B-Instruct was also evaluated without the adapter under identical conditions: the same 4-bit base checkpoint, chat template, prompt, schema context, retrieved exemplars, greedy decoding (at most 256 new tokens), and scoring. Both models were run in the same session, so they differed only in the presence of the adapter, and the fine-tuned model reproduced its reported results exactly.

The database-construction code, benchmark materials, retrieval exemplar store, and fine-tuning data-generation code are available in the Figshare reproducibility package (see Supplementary Materials).

5. Results

This section reports the evaluation on the 48-question benchmark of Section 4.1. Execution accuracy is the primary metric: a generated query is correct when its executed result set matches that of the reference query, compared as an order-insensitive multiset with floating-point values rounded. Fidelity (the Jaccard overlap of result rows) and an outcome category (exact match, value mismatch, SQL error, empty result, or no prediction) support the failure analysis.

5.1. Effect of Retrieval Grounding

Table 6 reports execution accuracy for the two locally deployable models under the three context configurations. Supplying controlled vocabularies produced no meaningful improvement over the bare schema (one additional question for the general model, none for the code-specialized model). Retrieving validated worked examples, by contrast, approximately doubled the execution accuracy for both models, from 23–25% to 46–48%. The improvement was statistically significant and free of regressions: a paired McNemar test on per-question correctness identified eleven questions that changed from incorrect to correct with example retrieval and none that regressed, for each model (exact two-sided p approximately 0.001). The two retrieval conditions thus differed in kind rather than degree: vocabulary lists indicate which entities exist, whereas worked examples demonstrate how to compose them, and only the latter affected accuracy on this schema.

Table 6. Execution accuracy by context configuration (out of 48 questions).

Configuration	Qwen3-30B-A3B (General)	Qwen3-Coder-30B-A3B
Schema only	11 (23%)	12 (25%)
Schema + vocabulary	12 (25%)	12 (25%)
Schema + retrieved examples	22 (46%)	23 (48%)

5.2. Performance by Query Complexity

Table 7 decomposes the schema-only and retrieval configurations by complexity tier for the code-specialized model; the general model followed the same pattern. The benefit of retrieval was concentrated at the intermediate tier, where accuracy rose from 27% to 80%, recovering most two- and three-table join queries. Single-table queries (T1) were already handled moderately well and improved slightly, whereas the complex multi-join queries (T3) remained near the floor, rising only from zero to two of twenty. Accuracy therefore degraded sharply with structural complexity, and the residual difficulty was almost entirely confined to the most complex queries. The retrieval configuration answered 12 of the 15 intermediate-tier questions, which link a batch to its product, material lots, or process features, whereas its overall accuracy of 46–48% was held down by the hardest analytical queries (T3), the cases that most require human review.

Table 7. Execution accuracy by complexity tier (Qwen3-Coder-30B-A3B).

Tier	Schema Only	Schema + Retrieved Examples
T1 (n = 13)	8 (62%)	9 (69%)
T2 (n = 15)	4 (27%)	12 (80%)
T3 (n = 20)	0 (0%)	2 (10%)

5.3. Model Specialization and Additional Refinements

Across all configurations, the code-specialized model did not significantly outperform the general-purpose model of the same architecture and scale (48% versus 46% with retrieved examples; 25% versus 23% with the schema alone), and a paired McNemar test found no significant difference in either condition ($p = 1.0$). For natural-language-to-SQL over this schema, retrieval grounding rather than code specialization determined accuracy.

Two refinements were evaluated as additions to the retrieval configuration, and neither improved accuracy. Adding the curated analytical views reduced accuracy from 46–48% to 42% for both models. The mechanism was visible in the generated queries: the models adopted the convenience columns but frequently applied them incorrectly, selecting a column that was null for the requested metric or mismatching a view to a question, which raised the SQL-error rate in the schema-only condition to 19%, while the additional column names enlarged the prompt without commensurate benefit. Denormalizing the schema into wide reporting views was therefore counterproductive. Layering execution-guided self-consistency on the view-augmented configuration (five sampled candidates with up to two repair rounds for the code-specialized model and three candidates with one repair round for the general model) produced at most a negligible gain: accuracy changed from 20 to 21 of 48 for the code-specialized model, with one question changing from incorrect to correct and none from correct to incorrect (exact McNemar $p = 1.0$), and remained at 20 of 48 for the general model, with no question changing in either direction, while median per-query generation time increased three- to fivefold. Self-consistency and execution repair help only when failures are non-executable or inconsistent across samples, whereas the dominant residual failures here were executable and stable (Section 5.5), so the majority vote tended to agree on an incorrect result.

The confidence score produced by the voting procedure was, in both models, weakly but positively associated with correctness. Correct predictions carried higher mean confidence than incorrect ones (0.98 versus 0.88 for the code-specialized model; 1.00 versus 0.93 for the general model), and the point-biserial correlation between confidence and correctness was 0.36 and 0.22, respectively. However, both models assigned the maximum confidence to a large majority of questions, including many that were wrong, so the signal is usable as a soft indicator but does not by itself constitute a reliable correctness filter. This supports retaining human validation of low-confidence outputs while cautioning against treating model confidence as a guarantee.

5.4. Schema-Specialized Fine-Tuning

Table 8 reports the fine-tuned seven-billion-parameter model in two configurations, evaluated with the schema-only context it was trained on and with retrieved examples added at inference, against its untuned base model and the strongest off-the-shelf configurations. Relative to the untuned base evaluated under identical conditions, fine-tuning raised overall accuracy from 9 to 21 of 48 (19% to 44%) with schema-only context and from 14 to 22 of 48 (29% to 46%) with retrieved examples. In paired comparisons, 14 questions changed from incorrect to correct, and 2 from correct to incorrect, in the first condition (exact McNemar $p = 0.004$), and 11 and 3 in the second ($p = 0.057$). On the most complex queries, the untuned model answered none and one of twenty, against four and five after fine-tuning, with no T3 question lost in either condition; this direction is consistent with the objective of the T3-weighted training set, although with twenty questions, the tier-level difference is not statistically significant ($p = 0.125$ in each condition). With retrieval, the fine-tuned model reached five of twenty (25%), the highest T3 accuracy observed among the configurations evaluated. Because the untuned model with retrieval (29%) remained well below the 30B-A3B model with retrieval (48%), the parity of the fine-tuned model with the larger model (discussed below) is attributable to adaptation rather than to the smaller base model. The deliberately T3-weighted training set thus improved performance precisely where the system had previously failed most, and the two interventions proved complementary: relative to the tuned model alone, adding retrieval recovered intermediate-tier accuracy (T2 from 9 to 11 of 15) while preserving the T3 gain (T3 from 4 to 5).

Table 8. Execution accuracy across the principal configurations (out of 48), with per-tier detail. The first three rows are the 30B-A3B code-specialized model; the next two are the untuned Qwen2.5-Coder-7B (control), and the last two are the fine-tuned Qwen2.5-Coder-7B.

Configuration	Overall	T1 (13)	T2 (15)	T3 (20)
Schema only (30B)	12 (25%)	8	4	0
Schema + retrieved examples (30B)	23 (48%)	9	12	2
Retrieval + views + self-consistency (30B)	21 (44%)	9	10	2
Untuned 7B (schema only)	9 (19%)	7	2	0
Untuned 7B + retrieved examples	14 (29%)	7	6	1
Fine-tuned 7B (schema only)	21 (44%)	8	9	4
Fine-tuned 7B + retrieved examples	22 (46%)	6	11	5

Figure 3 visualizes these configurations across the three complexity tiers. The fine-tuned model did not exceed the 30B-A3B retrieval configuration in overall accuracy (46% versus 48%).

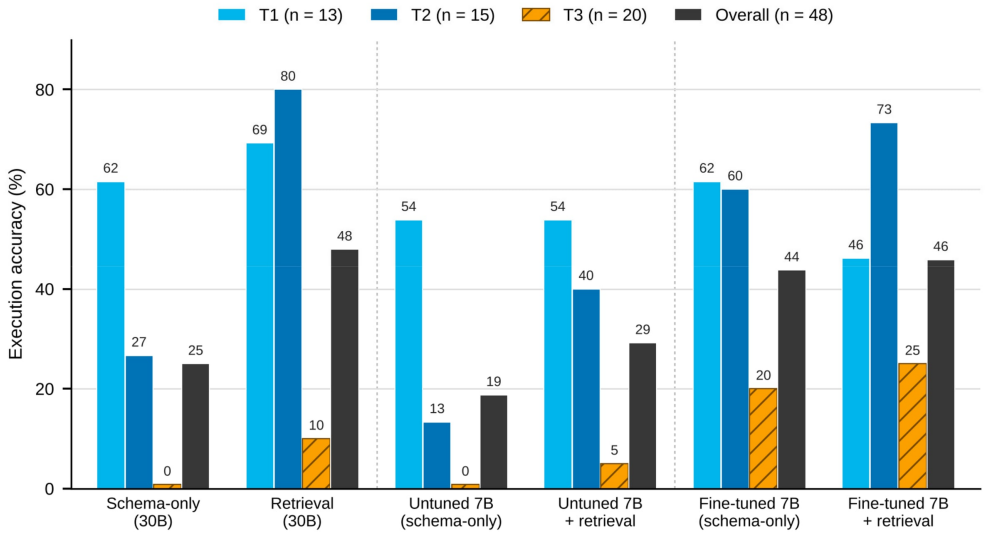


Figure 3. Execution accuracy by configuration (schema-only, retrieval, and fine-tuned variants, with the untuned 7B model as control) and by complexity tier.

Two observations qualify this parity in the tuned model's favor. First, it is a dense model with 7.6 billion parameters, roughly a quarter of the 30.5 billion total parameters of the 30B-A3B model, although all of its parameters are active for each token, compared with about 3.3 billion active parameters in the mixture-of-experts model. In 4-bit precision, its weights occupy about 5.5 GB, against about 18 GB for the 30B-A3B model (Section 4.5), so matching the larger model overall while surpassing it on the hardest tier is a favorable trade-off in memory footprint for single-GPU on-premise deployment. Second, the modest overall figure masks a redistribution: adding retrieval to the tuned model lowered single-table accuracy (T1 from 8 to 6 of 13), as the worked examples appear to distract the model on trivial queries, offsetting part of the gains at the higher tiers. The strongest configuration for complex analytical queries is therefore the fine-tuned model with retrieval, while the

highest overall accuracy remains the larger model with retrieval; which to prefer depends on the query mix and the deployment constraints.

5.5. Failure Analysis

Categorizing every outcome clarified the accuracy ceiling. Table 9 gives the number of questions in each outcome category for each configuration. In the schema-only configuration, the errors were dominated by two patterns rooted in the normalized schema: queries that omitted the stage and statistic filters required by the entity-attribute-value design of the quality table (the single largest cause, accounting for the entire set of measurement questions that require such filters), and queries that assumed a flatter schema than exists—for example, treating an attribute stored in the characteristic catalog as a direct column. Retrieving worked examples that demonstrate these patterns removed most of them, which is the source of the intermediate-tier recovery in Section 5.2. After retrieval, the residual failures were concentrated in a single category: value mismatches accounted for roughly twenty of the forty-eight questions for both models, while empty results and SQL errors each fell to five or fewer. These were queries that were syntactically valid and executed without error, and returned a plausible but incorrect result set, typically through an incorrect aggregation grain, a missing or over-broad join predicate, or a misreading of the intended grouping. Such errors are, by construction, invisible to execution-guided methods, because the database raises no error and independently sampled candidates tend to repeat the same misreading. The ceiling on this benchmark is therefore one of semantic rather than syntactic correctness, most pronounced on the complex queries that require multi-step compositional reasoning; retrieval raised T3 accuracy only from zero to two of twenty, and only the fine-tuned configurations exceeded that level (four and five of twenty).

Table 9. Outcome categories by configuration (number of questions out of 48). For the schema-only and retrieved-example configurations of the 30B-A3B models (Table 6), executed queries with an incorrect result and queries that did not execute are reported as combined counts.

Configuration	Match	Executed, Incorrect Result		Not Executed	
		Value Mismatch	Empty Result	SQL Error	No Prediction
Qwen3-30B-A3B (general)					
Schema only	11	35		2	
Schema + vocabulary	12	23	5	8	0
Schema + retrieved examples	22	25		1	
Retrieval + views	20	20	6	2	0
Retrieval + views + self-consistency	20	19	8	1	0
Qwen3-Coder-30B-A3B					
Schema only	12	36		0	
Schema + vocabulary	12	25	7	4	0
Schema + retrieved examples	23	25		0	
Retrieval + views	20	22	5	1	0
Retrieval + views + self-consistency	21	22	4	0	1

Table 9. *Cont.*

Configuration	Match	Executed, Incorrect Result		Not Executed	
		Value Mismatch	Empty Result	SQL Error	No Prediction
Qwen2.5-Coder-7B					
Untuned, schema only	9	22	4	13	0
Untuned + retrieved examples	14	23	6	5	0
Fine-tuned, schema only	21	17	3	7	0
Fine-tuned + retrieved examples	22	21	2	3	0

Recent text-to-SQL studies indicate that low-to-moderate execution accuracy is common once models are evaluated beyond simplified academic benchmarks. On BIRD, even GPT-4 reached only 54.89% execution accuracy against 92.96% human performance [18]. At the enterprise-workflow level, Spider 2.0 makes the difficulty starker still: the best agentic configuration solved only 21.36% of tasks, and traditional text-to-SQL methods dropped below 6% execution accuracy on Spider 2.0-Lite despite much higher results on Spider 1.0 and BIRD [20]. Even ReFoRCE, a state-of-the-art agentic framework using schema compression, self-refinement, consensus voting, and column exploration, achieved only 35.83% and 36.56% execution accuracy on Spider 2.0-Snow and Spider 2.0-Lite respectively [38]. The semantic ceiling observed here is therefore in line with the difficulty these benchmarks report.

6. Discussion

The evaluation yields a coherent and, in places, deliberately negative account of what does and does not improve natural-language-to-SQL on a realistic, ISA-88-aligned manufacturing database. The single largest lever was retrieval of validated query examples, which approximately doubled execution accuracy with a significant effect and no question losing accuracy for either of the two evaluated models on the 48-question benchmark, and recovered most intermediate-complexity queries. Controlled vocabularies, code specialization, denormalized analytical views, and execution-guided self-consistency each added little or nothing, the last confirmed across both models, and the analytical views were mildly counterproductive. Targeted fine-tuning of a small model on a compositional, execution-validated synthetic dataset more than doubled the accuracy of the untuned base model under identical conditions (from 19% to 44%), was the only intervention to raise accuracy on the hardest tier above two of twenty, and it was complementary to retrieval: the fine-tuned seven-billion-parameter model combined with retrieved examples achieved the highest observed accuracy on complex analytical queries of any configuration while matching the much larger model overall, a favorable result for on-premise deployment even though absolute accuracy on the hardest queries remained limited. These findings are consistent with the broader text-to-SQL literature, in which accuracy on realistic enterprise schemas remains well below that on simpler academic benchmarks (Section 5.5), in which retrieval and in-context demonstrations are strong contributors, and in which locally deployable open-weight models trail frontier systems while remaining attractive where data cannot leave the premises; the limitations observed on the present ISA-88 benchmark are thus consistent with broader challenges in realistic deployment, particularly when schemas are large, normalized, and domain-specific, and require multi-step analytical reasoning.

For practice, the results suggest a pragmatic on-premise configuration: a single-GPU open-weight model grounded with retrieved validated examples answered 12 of the 15 intermediate-tier (two- and three-table) benchmark questions with fully auditable behavior, even though overall accuracy (46–48%) is held down by the hardest analytical queries, and a small fine-tuned model extends reach on those complex queries while remaining inexpensive to host. Relative to the closest industrial neighbors, which provide natural-language access over sensor streams [27] or simulation output via knowledge graphs [28], the contribution here is executable SQL over hierarchical relational batch records with execution-based fidelity checking. The weak association between the model's confidence signal and correctness reinforces the design choice to keep a human in the loop: confidence is useful for triage but is not a substitute for validation, and the execution-based comparison against ground truth, rather than the model's self-assessment, is what provides an objective correctness signal in the offline evaluation.

7. Limitations and Future Work

Several limitations bound the claims of this study and should be read as design boundaries. The benchmark comprises 48 questions over a single production line, so absolute accuracies should be read as indicative rather than definitive, and broader benchmarks across plants and processes are needed. The fine-tuning data were synthetic; although every query was validated by execution and the human-curated benchmark was held out, the training distribution may not fully reflect the questions practitioners ask, and overfitting to the synthetic distribution remains a risk that a larger and more varied training set, or additional human-authored evaluation items, would mitigate. The fine-tuning effect was isolated against the untuned base model on the same 48 questions; with twenty T3 questions, the tier-level gain is consistent in direction but not statistically significant, so larger evaluation sets are needed to estimate it precisely. Accuracy was measured with hosted inference of the published open weights and on a cloud-hosted GPU rather than on plant infrastructure; a local 4-bit deployment of the Qwen3 models may change individual outputs, and the memory footprints given are estimates, so the latency, memory use, and throughput of an on-premise deployment remain to be measured on the target hardware.

Future work follows from these boundaries. The accuracy ceiling is semantic rather than syntactic, and closing it on complex compositional queries will require approaches beyond context engineering: richer and larger fine-tuning datasets, larger schema-adapted models, and query-decomposition or plan-then-generate strategies that target multi-step reasoning. Schema linking, which supplies fewer question-relevant columns rather than more, is a promising alternative to the denormalized views that proved counterproductive here. Validation on additional plants and processes would further strengthen external validity.

8. Conclusions

This study implemented and evaluated a locally deployable, schema-aware, RAG-grounded LLM system that translates natural-language questions into SQL over an ISA-88-aligned batch-manufacturing database, with confidence scoring, and designed for human-in-the-loop validation. On a tiered 48-question benchmark, retrieving validated query examples was the decisive intervention, approximately doubling execution accuracy to 46–48% with a significant effect and no regressions for the two evaluated models on the 48-question benchmark, while code specialization, denormalized analytical views and execution-guided self-consistency added little; a small code model fine-tuned on a tier-balanced, execution-validated synthetic dataset more than doubled the accuracy of its untuned base, was the only intervention to raise accuracy on the most complex queries

above two of twenty, and, combined with retrieval, achieved the highest observed accuracy on them, while a seven-billion-parameter model matched a far larger one overall. Rather than competing with cloud frontier systems on accuracy, the results show that a deliberately lightweight, auditable pipeline built on locally deployable open-weight models is feasible for regulated batch-manufacturing analytics, and it maps the query classes for which human review remains essential. The approach offers lessons for other regulated, data-rich industrial domains where data sovereignty and validatability outweigh raw model performance.

Supplementary Materials: The following supporting information can be downloaded at: https://www.mdpi.com/article/10.3390/app162110971/s1, File S1: README describing the reproducibility package and step-by-step instructions for reproducing the experiments; File S2: reproducibility package (ZIP) containing the code that constructs the ISA-88-aligned database, the 48 benchmark questions with their complexity tiers, reference SQL queries and reference result sets, the retrieval exemplar store and retriever, and the code used to generate the fine-tuning data.

Author Contributions: Conceptualization, L.R., J.Z.-A., J.V., M.U., G.M., S.D., and E.G.-F.; methodology, L.R. and J.Z.-A.; validation, J.V. and E.G.-F.; formal analysis, J.Z.-A.; investigation, J.Z.-A., M.U., G.M., and S.D.; writing—original draft preparation, L.R., J.Z.-A., M.U., G.M., and S.D.; writing—review and editing, F.A., J.V., and E.G.-F.; visualization, J.Z.-A.; project administration, L.R.; funding acquisition, F.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Ongoing Research Funding Program (ORF-2026-1514), King Saud University, Riyadh, Saudi Arabia.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The pharmaceutical tablet-manufacturing data used in this study were published by Žagar and Mihelič [3] and are publicly available in Figshare at https://doi.org/10.6084/m9.figshare.c.5645578.v1. The ISA-88-aligned relational database used for the analyses was derived from these data. The materials needed to reproduce the experiments are provided as Supplementary Materials and are also openly available at https://doi.org/10.6084/m9.figshare.34868340. The 48 benchmark questions are also listed in Table A3.

Conflicts of Interest: Author Jose Vazquez was employed by Altra Systems Inc. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest. The Altra Systems Inc had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Appendix A. Prompt Templates and Configuration

Appendix A.1. Prompt Structure

Each query was generated from a single user prompt comprising three parts: a schema description (every table with its columns and foreign keys, the analytical views, and controlled domain vocabularies), a task instruction, and the question. In the retrieval condition, k = 3 validated worked examples (a natural-language question paired with its reference SQL), retrieved from a held-out store, were inserted before the task instruction. The template skeleton is as follows.

```
# Database schema (SQLite)
TABLE batch(batch_id, product_code, campaign_id, batch_size, …)
… all tables with columns and foreign keys, plus the analytical views …
# Domain hints (controlled vocabularies)
characteristic.name in: dissolution, hardness, thickness, …
```

```
batch_quality.stage in: core, fct statistic in: min, max, av, rsd, single
# Worked examples (retrieval condition only; k = 3, held out)
Q: <natural-language question>
SQL: <validated reference query>
# Task
Write ONE SQLite query that answers the question. Use only the tables and
columns above. Return only the SQL statement.
Question: <user question>
SQL:
```

Appendix A.2. Models

The two primary models were Qwen3-30B-A3B-Instruct and Qwen3-Coder-30B-A3B-Instruct (mixture-of-experts, 30.5 billion total and about 3.3 billion active parameters); the fine-tuned model was Qwen2.5-Coder-7B-Instruct. The exact model identifiers were Qwen3-30B-A3B-Instruct-2507 and Qwen3-Coder-30B-A3B-Instruct, accessed on June 2026 through the OpenRouter API, an OpenAI-compatible chat-completions endpoint serving the published open weights at the precision chosen by the provider, and Qwen2.5-Coder-7B-Instruct, used through the 4-bit checkpoint unsloth/Qwen2.5-Coder-7B-Instruct-bnb-4bit (revision 4858886) for both fine-tuning and evaluation. For the Qwen3 models, each request contained the single prompt of Appendix A.1 and was decoded at temperature 0 with the provider's default output-length limit, and the generated SQL was executed locally against the SQLite database. Fine-tuning was performed with Unsloth and the TRL supervised fine-tuning trainer in a Kaggle notebook. The 7B models were loaded in 4-bit precision (bitsandbytes) with a maximum sequence length of 2048 tokens and decoded greedily with at most 256 new tokens; they were evaluated with PyTorch 2.11.0 (CUDA 12.8), Transformers 5.16.1, Unsloth 2026.10.10, PEFT 0.20.0, TRL 1.15.0, and bitsandbytes 0.50.2 on a single NVIDIA Tesla T4 GPU (16 GB) in a Kaggle notebook; the evaluation harness uses Python 3.13.14 and SQLite 3.53.1.

Appendix A.3. Retrieval

Worked examples were retrieved by a lightweight, dependency-free lexical-similarity ranker (token overlap weighted by inverse document frequency) over a held-out exemplar store disjoint from the 48 benchmark questions; the three highest-ranked examples were supplied per question. The exemplar store contains 19 exemplars, none of which is a benchmark question; the retrieval configurations of the Qwen3 models without analytical views used the first 12 exemplars, and all other retrieval configurations used the full store.

Appendix A.4. Fine-Tuning

Adaptation used QLoRA [48] via the Unsloth library [49]: the base model was loaded in 4-bit precision, and low-rank adapters of rank 16 (alpha 16, no dropout) were trained on the attention and feed-forward projection matrices, with the loss computed only on the SQL completion. Training ran for two epochs at learning rate 2×10^{-4} with a linear schedule, an effective batch size of eight (per-device batch size two, gradient accumulation four), the adamw_8bit optimizer and a maximum sequence length of 2048, on a single 16 GB GPU. The training set was synthetic and execution-validated, tier-balanced to approximately 10/35/55 across T1/T2/T3, with language-model paraphrasing (Qwen3-30B-A3B-Instruct-2507, temperature 0.8) for natural-language diversity, and questions matching the benchmark or the retrieval exemplars excluded. The data-generation seed was 0, and the training and LoRA initialization seed was 3407.

Appendix A.5. Execution-Guided Self-Consistency

Where evaluated, five candidates for the code-specialized model and three for the general model were sampled (the first at temperature zero, the remainder at temperature 0.6 without a fixed sampling seed); each was executed, and candidates that raised an error or returned no rows were repaired by returning the execution feedback to the model for up to two rounds (code-specialized model) or one round (general model); the answer was selected by majority vote over executed result sets, and the fraction of candidates agreeing on the selected result was reported as a confidence score.

Appendix A.6. Metrics

Execution accuracy counts a query correct when its executed result set equals that of the reference query, compared as an order- and column-order-insensitive multiset with floating-point values rounded to four decimal places. Fidelity is the Jaccard overlap of result rows. Each outcome is categorized as exact match, value mismatch, SQL error, empty result, or no prediction. Paired configurations are compared with an exact two-sided McNemar test on per-question correctness. Table A1 summarizes the key configuration parameters.

Table A1. Key configuration parameters.

Component	Setting
General/coder models	Qwen3-30B-A3B-Instruct; Qwen3-Coder-30B-A3B-Instruct
Fine-tuned model	Qwen2.5-Coder-7B-Instruct (QLoRA adapter)
Quantization	4-bit for the 7B models; precision of the hosted service for the Qwen3 models (4-bit for single-GPU deployment)
Quantization (deployment)	4-bit, single workstation-class GPU
Decoding (single-shot)	temperature 0 (deterministic)
Retrieval	lexical similarity, top-3 held-out examples
LoRA rank/alpha/dropout	16/16/0
LoRA target modules	q, k, v, o, gate, up, down projections
Epochs/learning rate/schedule	$2/2 \times 10^{-4}$/linear
Effective batch/max sequence length	8/2048
Self-consistency	5 samples and up to 2 repair rounds (code-specialized model); 3 samples and 1 repair round (general model); majority vote
Benchmark	48 questions, three tiers (13/15/20)

Appendix B. Database Schema

Appendix B.1. Relational Schema

The relational schema is organized into the ISA-88 sub-models, with the batch table as the central hub through which almost every join passes. Figure A1 shows the sub-models and their relationship to the hub, and Table A2 lists the sixteen tables with their primary keys and roles. The product-and-recipe, physical-equipment, and procedural-execution layers are modeled in alignment with ISA-88 from domain knowledge, whereas the material-genealogy, process-data, and quality layers are populated directly from the source dataset. Quality results are stored in long (entity-attribute-value) form keyed by a

shared characteristic catalog, with the stage and statistic columns distinguishing core from film-coated measurements and identifying the reported summary statistic; convenience views such as v_batch_quality_wide pivot this table into one row per batch for readability (Section 5). Full data-definition statements are provided with the released code.

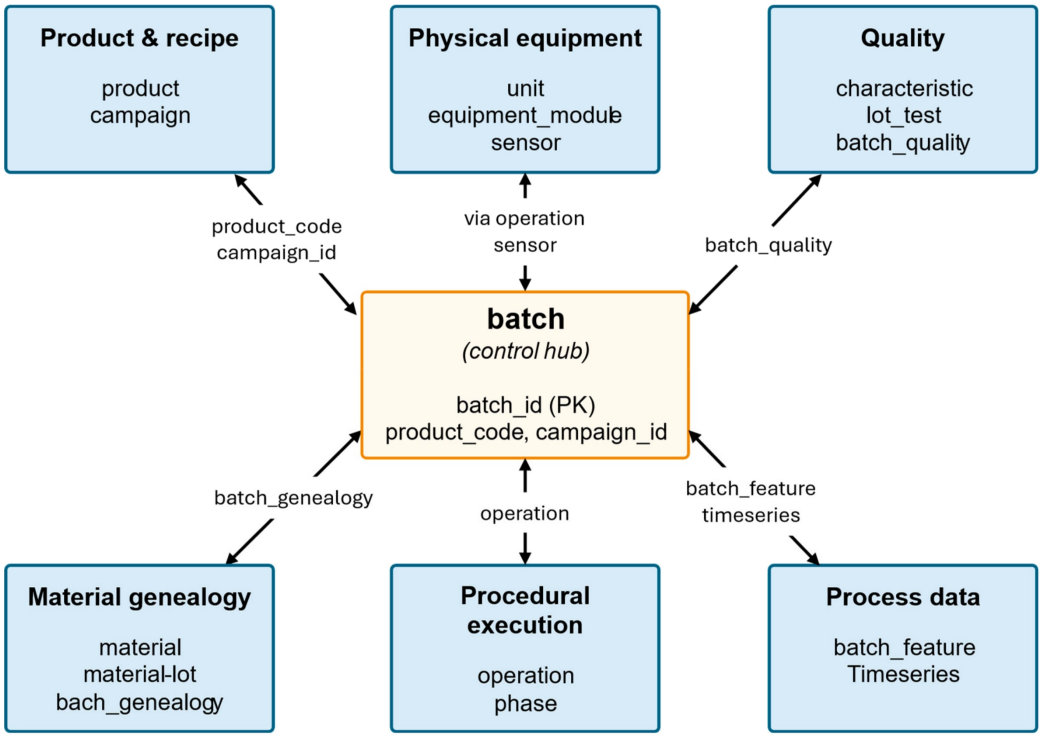


Figure A1. ISA-88-aligned database schema. The six sub-models (product and recipe, physical equipment, material genealogy, procedural execution, process data, and quality) are organized around the central batch table, through which almost every join passes.

Table A2. Tables of the ISA-88-aligned schema, by sub-model.

ISA-88 Sub-Model	Table	Primary Key	Role and Key Relationships
Product and recipe	product	product_code	Product sub-family: strength, batch size and normalization factor.
	campaign	campaign_id	Plant-wide production campaign, derived from the time series.
Central hub	batch	batch_id	One manufactured batch; links to product and campaign and is referenced by the genealogy, quality, procedural and process tables.
Physical equipment	unit	unit_id	Process unit (ISA-88 unit).
	equipment_module	module_id	Equipment module within a unit (FK unit_id).
	sensor	sensor_id	Measured parameter on a module (FK module_id); parameter_code matches a time-series column.
Material genealogy	material	material_id	Material type (API, SMCC, lactose, starch).
	material_lot	lot_id	Incoming lot of a material (FK material_id).
	batch_genealogy	(batch_id, lot_id)	Which lots a batch consumed (FK batch_id, FK lot_id).

Table A2. *Cont.*

ISA-88 Sub-Model	Table	Primary Key	Role and Key Relationships
Quality	characteristic	characteristic_id	Catalog of measured properties (applies_to raw_material, intermediate or final); shared by lot_test and batch_quality.
	lot_test	lot_test_id	Incoming-material test result (FK lot_id, FK characteristic_id).
	batch_quality	result_id	Intermediate and final product results in long form (FK batch_id, FK characteristic_id; stage, statistic, value).
Procedural execution	operation	operation_id	Operation on a batch at a unit (FK batch_id, FK unit_id), e.g., compression.
	phase	phase_id	Phase within an operation (FK operation_id): start-up or production.
Process data	batch_feature	feature_id	Per-batch process feature (FK batch_id, optional FK phase_id; feature_name, value).
	timeseries	reading_id	Ten-second compression sensor readings (FK batch_id); wide columns, long form via a view.

Appendix B.2. Benchmark Question Bank

Table A3 lists the 48 natural-language questions that constitute the evaluation benchmark introduced in Section 4.1. Each question is assigned to one of three complexity tiers, reflecting the structural complexity of its reference (gold) SQL rather than the surface form of the question: single-table or simple-aggregation queries (T1, n = 13); two- to three-table join queries (T2, n = 15); and multi-join queries involving aggregation, filtering, temporal logic, or sub-queries (T3, n = 20). Each question is additionally tagged with the data theme it targets, following the breakdown in Section 4.1: genealogy (product, recipe, batch, and material-lot structure and lineage; 14 questions), quality (incoming raw-material and intermediate or final product quality measurements; 14), process (compression-process features such as press speed, compression force, ejection, and waste; 5), equipment/sensor (equipment-module and sensor relationships and readings; 3), campaign (campaign-level roll-ups; 6), and temporal (analysis by weekday/weekend, month or year; 6). Each question is paired with an execution-validated reference query as described in Section 4.2; the questions are reproduced here, and the reference SQL and result sets accompany the released benchmark.

Table A3. The 48-question evaluation benchmark, grouped by complexity tier, with data-theme category.

ID	Tier	Main Category	Natural-Language Question
q01	T1	Genealogy	How many batches were produced for each product code?
q02	T1	Quality	What is the overall average final-product dissolution?
q03	T1	Genealogy	List each product strength with its batch size and normalization factor.

Table A3. *Cont.*

ID	Tier	Main Category	Natural-Language Question
q04	T1	Temporal	How many batches were run on weekends versus weekdays?
q05	T1	Campaign	How many distinct campaigns are in the dataset?
q19	T1	Genealogy	How many distinct product strengths are there?
q20	T1	Genealogy	What is the total number of tablets produced across all batches?
q21	T1	Quality	What are the minimum and maximum final-product dissolution values recorded?
q22	T1	Genealogy	List product codes with their normalization factors, highest first.
q23	T1	Genealogy	How many material lots are recorded in total?
q24	T1	Quality	How many batches had a final batch yield below 90%?
q25	T1	Process	What is the average production tablet-press speed across all batches?
q26	T1	Campaign	How many batches are assigned to a campaign?
q06	T2	Quality	For each product strength, what is the average final tablet hardness?
q07	T2	Quality	Which five lactose lots had the highest water content?
q08	T2	Genealogy	How many batches consumed API lot number 170?
q09	T2	Quality	List the ten batches with the lowest final dissolution, with their product code.
q10	T2	Genealogy	For each material, how many lots exist and how many batches consumed them?
q27	T2	Equip-ment/sensor	How many sensors are mapped to each equipment module?
q28	T2	Genealogy	How many batches were produced per product strength?
q29	T2	Quality	Which five API lots have the highest assay content?
q30	T2	Quality	For each product code, what is the average final tablet tensile strength?
q31	T2	Quality	Which five SMCC lots have the lowest bulk density?
q32	T2	Campaign	Which ten campaigns contained the most batches?
q33	T2	Genealogy	How many distinct API lots did each product code use?

Table A3. *Cont.*

ID	Tier	Main Category	Natural-Language Question
q34	T2	Process	List the ten batches with the highest start-up waste, with their product code.
q35	T2	Quality	What is the average final residual solvent per product strength?
q36	T2	Quality	What is the average API assay content per product strength?
q11	T3	Genealogy	Which API lots are associated with the lowest average final dissolution (lots used in at least five batches)?
q12	T3	Process	What is the mean production-phase main compression force per product strength?
q13	T3	Campaign	Per campaign, how many batches exceeded 4000 start-up rejected tablets, and what was their average final total-impurity content?
q14	T3	Genealogy	For batches whose tablet-core average hardness was below 50 N, what was the assay content of the API lots they used?
q15	T3	Temporal	What was the average production tablet-press speed by month of manufacture?
q16	T3	Quality	Which batches had a final dissolution below 90%, and in which campaign did they occur?
q17	T3	Equip-ment/sensor	For the first batch of product code 1, what was the average ejection force recorded by the ejection-station sensor, and on which equipment module?
q18	T3	Temporal	Compare average final tablet tensile strength for weekend versus weekday batches, per product strength.
q37	T3	Process	Per product strength, give the average production main-compression force and the average final dissolution.
q38	T3	Genealogy	Which five API lots were used in the most batches, and what was their average assay content?
q39	T3	Temporal	For batches manufactured in 2020, what was the average final dissolution by month?
q40	T3	Temporal	Compare average final tablet hardness for weekend versus weekday batches, per product strength.
q41	T3	Campaign	Which campaigns had an average final total-impurity content above the overall average?
q42	T3	Quality	For each product strength, what fraction of batches had a final dissolution of at least 95%?

Table A3. *Cont.*

ID	Tier	Main Category	Natural-Language Question
q43	T3	Genealogy	List batches whose final dissolution was below 85%, with the assay content of the API lot they used.
q44	T3	Process	Rank product strengths by their average maximum ejection force during production.
q45	T3	Equipment/sensor	For the first batch of product code 1, give the average value recorded by each sensor and the module it sits on.
q46	T3	Temporal	How many batches were produced for each combination of product strength and weekend/weekday?
q47	T3	Campaign	Rank the ten campaigns that produced the most tablets in total.
q48	T3	Quality	For each product strength, compare the average tablet-core hardness with the average film-coated hardness.

References

1. Tao, F.; Qi, Q.; Liu, A.; Kusiak, A. Data-Driven Smart Manufacturing. *J. Manuf. Syst.* **2018**, *48*, 157–169. [CrossRef]
2. Arden, N.S.; Fisher, A.C.; Tyner, K.; Yu, L.X.; Lee, S.L.; Kopcha, M. Industry 4.0 for Pharmaceutical Manufacturing: Preparing for the Smart Factories of the Future. *Int. J. Pharm.* **2021**, *602*, 120554. [CrossRef] [PubMed]
3. Žagar, J.; Mihelič, J. Big Data Collection in Pharmaceutical Manufacturing and Its Use for Product Quality Predictions. *Sci. Data* **2022**, *9*, 99. [CrossRef] [PubMed]
4. Affolter, K.; Stockinger, K.; Bernstein, A. A Comparative Survey of Recent Natural Language Interfaces for Databases. *VLDB J.* **2019**, *28*, 793–819. [CrossRef]
5. Jeon, J.; Sim, Y.; Lee, H.; Han, C.; Yun, D.; Kim, E.; Nagendra, S.L.; Jun, M.B.; Kim, Y.; Lee, S.W.; et al. ChatCNC: Conversational Machine Monitoring via Large Language Model and Real-Time Data Retrieval Augmented Generation. *J. Manuf. Syst.* **2025**, *79*, 504–514. [CrossRef]
6. Xia, L.; Li, C.; Zhang, C.; Liu, S.; Zheng, P. Leveraging Error-Assisted Fine-Tuning Large Language Models for Manufacturing Excellence. *Robot. Comput.-Integr. Manuf.* **2024**, *88*, 102728. [CrossRef]
7. Yao, Y.; Duan, J.; Xu, K.; Cai, Y.; Sun, Z.; Zhang, Y. A Survey on Large Language Model (LLM) Security and Privacy: The Good, the Bad, and the Ugly. *High-Confid. Comput.* **2024**, *4*, 100211. [CrossRef]
8. *ANSI/ISA-88.00.01-2010*; Batch Control Part 1: Models and Terminology. ISA: Research Triangle Park, NC, USA, 2010.
9. Virta, J.; Seilonen, I.; Tuomi, A.; Koskinen, K. SOA-Based Integration for Batch Process Management with OPC UA and ISA-88/95. In *Proceedings of the 15th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2010)*; IEEE: Piscataway, NJ, USA, 2010; pp. 1–8. [CrossRef]
10. Ji, Z.; Lee, N.; Frieske, R.; Yu, T.; Su, D.; Xu, Y.; Ishii, E.; Bang, Y.J.; Madotto, A.; Fung, P. Survey of Hallucination in Natural Language Generation. *ACM Comput. Surv.* **2023**, *55*, 1–38. [CrossRef]
11. Huang, L.; Yu, W.; Ma, W.; Zhong, W.; Feng, Z.; Wang, H.; Chen, Q.; Peng, W.; Feng, X.; Qin, B.; et al. A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions. *ACM Trans. Inf. Syst.* **2025**, *43*, 1–55. [CrossRef]
12. US FDA (Food and Drug Administration). *Data Integrity and Compliance with Drug CGMP: Questions and Answers*; Guidance for Industry; US Department of Health and Human Services: Silver Spring, MD, USA, 2018.
13. European Commission. *EudraLex Good Manufacturing Practice (GMP) Guidelines, Annex 11: Computerised Systems*; European Commission: Brussels, Belgium, 2011; Volume 4.
14. Kavasidis, I.; Lallas, E.; Leligkou, H.C.; Oikonomidis, G.; Karydas, D.; Gerogiannis, V.C.; Karageorgos, A. Deep Transformers for Computing and Predicting ALCOA+ Data Integrity Compliance in the Pharmaceutical Industry. *Appl. Sci.* **2023**, *13*, 7616. [CrossRef]
15. European Commission. *EudraLex Good Manufacturing Practice (GMP) Guidelines, Annex 22: Artificial Intelligence*; Consultation draft, July 7, 2025; European Commission, Directorate-General for Health and Food Safety: Brussels, Belgium, 2025; Volume 4.

16. Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Küttler, H.; Lewis, M.; Yih, W.-t.; Rocktäschel, T.; et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Advances in Neural Information Processing Systems 33*; Curran Associates: Red Hook, NY, USA, 2020; pp. 9459–9474.
17. Xu, X.; Lu, Y.; Vogel-Heuser, B.; Wang, L. Industry 4.0 and Industry 5.0—Inception, Conception and Perception. *J. Manuf. Syst.* **2021**, *61*, 530–535. [CrossRef]
18. Li, J.; Hui, B.; Qu, G.; Yang, J.; Li, B.; Li, B.; Wang, B.; Qin, B.; Geng, R.; Huo, N.; 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 36*; Curran Associates: Red Hook, NY, USA, 2023. [CrossRef]
19. Hong, Z.; Yuan, Z.; Zhang, Q.; Chen, H.; Dong, J.; Huang, F.; Huang, X. Next-Generation Database Interfaces: A Survey of LLM-Based Text-to-SQL. *IEEE Trans. Knowl. Data Eng.* **2025**, *37*, 7328–7345. [CrossRef]
20. Lei, F.; Chen, J.; Ye, Y.; Cao, R.; Shin, D.; Su, H.; Suo, Z.; Gao, H.; Hu, W.; Yin, P.; et al. Spider 2.0: Evaluating Language Models on Real-World Enterprise Text-to-SQL Workflows. In Proceedings of the Thirteenth International Conference on Learning Representations (ICLR 2025), Singapore, 24–28 April 2025. Available online: https://openreview.net/forum?id=XmProj9cPs (accessed on 2 November 2026).
21. Wang, X.; Wei, J.; Schuurmans, D.; Le, Q.; Chi, E.; Narang, S.; Chowdhery, A.; Zhou, D. Self-Consistency Improves Chain of Thought Reasoning in Language Models. In Proceedings of the Eleventh International Conference on Learning Representations (ICLR 2023), Kigali, Rwanda, 1–5 May 2023.
22. Salami, H.; Smith-Goettler, B.; Yadav, V. How Can Language Models Assist with Pharmaceuticals Manufacturing Deviations and Investigations? *Int. J. Pharm.* **2025**, *669*, 125100. [CrossRef] [PubMed]
23. Kadi, H.; Abdellatif, A.; Kemajou Njamen, D.I.; Pereme, F. Automatic Extraction of SmPC Document for IDMP Data Model Construction Using Foundation LLM and RAG. *Front. Med.* **2025**, *12*, 1598979. [CrossRef] [PubMed]
24. Kim, J.; Hur, M.; Min, M. From RAG to QA-RAG: Integrating Generative AI for Pharmaceutical Regulatory Compliance Process. In *Proceedings of the ACM Symposium on Applied Computing*; ACM: New York, NY, USA, 2025; pp. 1293–1295. [CrossRef]
25. Vieira-Vieira, C.H.; Kulkarni, S.S.; Zalewski, A.; Loffler, J.; Munch, J.; Kreuchwig, A. From Data Silos to Insights: The PRINCE Multi-Agent Knowledge Engine for Preclinical Drug Development. *Front. Artif. Intell.* **2025**, *8*, 1636809. [CrossRef] [PubMed]
26. Taskiran, N.P.; Tsai, C.E.J.; Huang, S.; Chakraborty, A.; Venkatasubramanian, V. A Knowledge-Graph-Based Pharmaceutical Engineering Chatbot for Drug Discovery. *Comput. Chem. Eng.* **2025**, *203*, 109318. [CrossRef]
27. González-Potes, A.; Martínez-Castro, D.; Paredes, C.M.; Ochoa-Brust, A.; Mena, L.J.; Martínez-Peláez, R.; Félix, V.G.; Félix-Cuadras, R.A. Hybrid AI and LLM-Enabled Agent-Based Real-Time Decision Support Architecture for Industrial Batch Processes: A Clean-in-Place Case Study. *AI* **2026**, *7*, 51. [CrossRef]
28. Thogaru, H.; Gopalakrishnan, S.; Ahmad, Z.; Deodhar, A. Intelligent Human-Machine Partnership for Manufacturing: Enhancing Warehouse Planning through Simulation-Driven Knowledge Graphs and LLM Collaboration. *arXiv* **2025**, arXiv:2512.18265. [CrossRef]
29. Wan, Y.; Chen, Z.; Liu, Y.; Chen, C.; Packianather, M. Empowering LLMs by Hybrid Retrieval-Augmented Generation for Domain-Centric Q&A in Smart Manufacturing. *Adv. Eng. Inform.* **2025**, *65*, 103212. [CrossRef]
30. Oyekan, J.; Turner, C.; Bax, M.; Graf, E. From Ontologies to Knowledge-Augmented Large Language Models for Automation: A Decision-Making Guidance for Achieving Human-Robot Collaboration in Industry 5.0. *Comput. Ind.* **2025**, *171*, 104329. [CrossRef]
31. Wang, T.; Zhang, B.; Jiang, D.; Li, D. A Multimodal Large Language Model Framework for Intelligent Perception and Decision-Making in Smart Manufacturing. *Sensors* **2025**, *25*, 3072. [CrossRef] [PubMed]
32. Mahmud, T.; Azharul Hasan, K.M.; Ahmed, M.; Chak, T.H.C. A Rule Based Approach for NLP Based Query Processing. In *Proceedings of the 2nd International Conference on Electrical Information and Communication Technologies (EICT 2015)*; IEEE: Piscataway, NJ, USA, 2016; pp. 78–82. [CrossRef]
33. Fan, W.; Ding, Y.; Ning, L.; Wang, S.; Li, H.; Yin, D.; Chua, T.-S.; Li, Q. A Survey on RAG Meeting LLMs: Towards Retrieval-Augmented Large Language Models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*; ACM: New York, NY, USA, 2024; pp. 6491–6501. [CrossRef]
34. Zhou, Q.; Zhou, Z.; Johnson, M.; Ngo, M.; Ferrari, F.; Ma, J. ClinicalRAG: Automating Pharmaceutical Label Quality Control with Hierarchical RAG and Large Language Models. *Proc. AAAI Conf. Artif. Intell.* **2025**, *39*, 29736–29738. [CrossRef]
35. Choi, H.; Jeong, J. A Conceptual Framework for a Latest Information-Maintaining Method Using Retrieval-Augmented Generation and a Large Language Model in Smart Manufacturing. *Machines* **2025**, *13*, 94. [CrossRef]
36. Mohammadjafari, A.; Maida, A.S.; Gottumukkala, R. From Natural Language to SQL: Review of LLM-Based Text-to-SQL Systems. *arXiv* **2025**, arXiv:2410.01066. [CrossRef]
37. Kumar, A.; Nagarkar, P.; Nalhe, P.; Vijayakumar, S. Deep Learning Driven Natural Language Text to SQL Query Conversion: A Survey. *arXiv* **2022**, arXiv:2208.04415. [CrossRef]

38. Deng, M.; Ramachandran, A.; Xu, C.; Hu, L.; Yao, Z.; Datta, A.; Zhang, H. ReFoRCE: A Text-to-SQL Agent with Self-Refinement, Consensus Enforcement, and Column Exploration. *arXiv* **2025**, arXiv:2502.00675. [CrossRef]
39. Li, B.; Zhang, J.; Fan, J.; Xu, Y.; Chen, C.; Tang, N.; Luo, Y. Alpha-SQL: Zero-Shot Text-to-SQL Using Monte Carlo Tree Search. In Proceedings of the 42nd International Conference on Machine Learning, PMLR, Vancouver, BC, Canada, 13–19 July 2025; Volume 267, pp. 36810–36830. Available online: https://proceedings.mlr.press/v267/li25dt.html (accessed on 2 November 2026).
40. Yan, L.; Wan, Q.; Liu, C.; Duan, S.; Han, P.; Xu, Y. SPS-SQL: Enhancing Text-to-SQL Generation on Small-Scale LLMs with Pre-Synthesized Queries. *Pattern Recognit. Lett.* **2025**, *196*, 45–51. [CrossRef]
41. Murali, T.; Anu Priya, A. Production Enhancement and Process Analysis for Pharmaceutical Industries. In *Proceedings of the 4th International Conference on Electronics and Sustainable Communication Systems (ICESC 2023)*; IEEE: Piscataway, NJ, USA, 2023; pp. 156–161. [CrossRef]
42. Vichev, S.; Marchev, A. RAGSQL: Context Retrieval Evaluation on Augmenting Text-to-SQL Prompts. In *Proceedings of the 2024 IEEE International Conference on Intelligent Systems (IS 2024)*; IEEE: Piscataway, NJ, USA, 2024; pp. 1–6. [CrossRef]
43. Dogruluk, E.; Acikgoz, H. Edge-Centric Federated Learning for LLMs in Smart Manufacturing: Architectures, Challenges, and Opportunities. In *Proceedings of the 4th International Conference on Innovative Mechanisms for Industry Applications (ICIMIA 2025)*; IEEE: Piscataway, NJ, USA, 2025; pp. 1250–1256. [CrossRef]
44. *Standard&'s IEC 62443-3-3:2013*; Industrial communication networks—Network and system security—Part 3-3: System security requirements and security levels. International Electrotechnical Commission: Geneva, Switzerland, 2013.
45. Qwen Team. Qwen3 Technical Report. *arXiv* **2025**, arXiv:2505.09388. [CrossRef]
46. Hui, B.; Yang, J.; Cui, Z.; Yang, J.; Liu, D.; Zhang, L.; Liu, T.; Zhang, J.; Yu, B.; Lu, K.; et al. Qwen2.5-Coder Technical Report. *arXiv* **2024**, arXiv:2409.12186. [CrossRef]
47. Jin, T.; Choi, Y.; Zhu, Y.; Kang, D. Text-to-SQL Benchmarks Are Broken: An In-Depth Analysis of Annotation Errors. In Proceedings of the 16th Annual Conference on Innovative Data Systems Research (CIDR '26), Chaminade, CA, USA, 18–21 January 2026. Available online: https://www.vldb.org/cidrdb/papers/2026/p5-jin.pdf (accessed on 2 November 2026).
48. Dettmers, T.; Pagnoni, A.; Holtzman, A.; Zettlemoyer, L. QLoRA: Efficient Finetuning of Quantized LLMs. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023)*; Curran Associates: Red Hook, NY, USA, 2023.
49. GitHub. Unsloth. Computer Software. 2024. Available online: https://github.com/unslothai/unsloth (accessed on 2 November 2026).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.