DataSpeck: An AI-Driven Human-in-the-Loop System for Automating Transformations in Data Conversion Workflows

Adil Rahman
University of Virginia
Charlottesville, VA, USA
adil@virginia.edu

Koichiro Niinuma
Fujitsu Research of America
Pittsburgh, PA, USA
kniinuma@fujitsu.com

Aakar Gupta
Fujitsu Research of America
Santa Clara, CA, USA
agupta@fujitsu.com

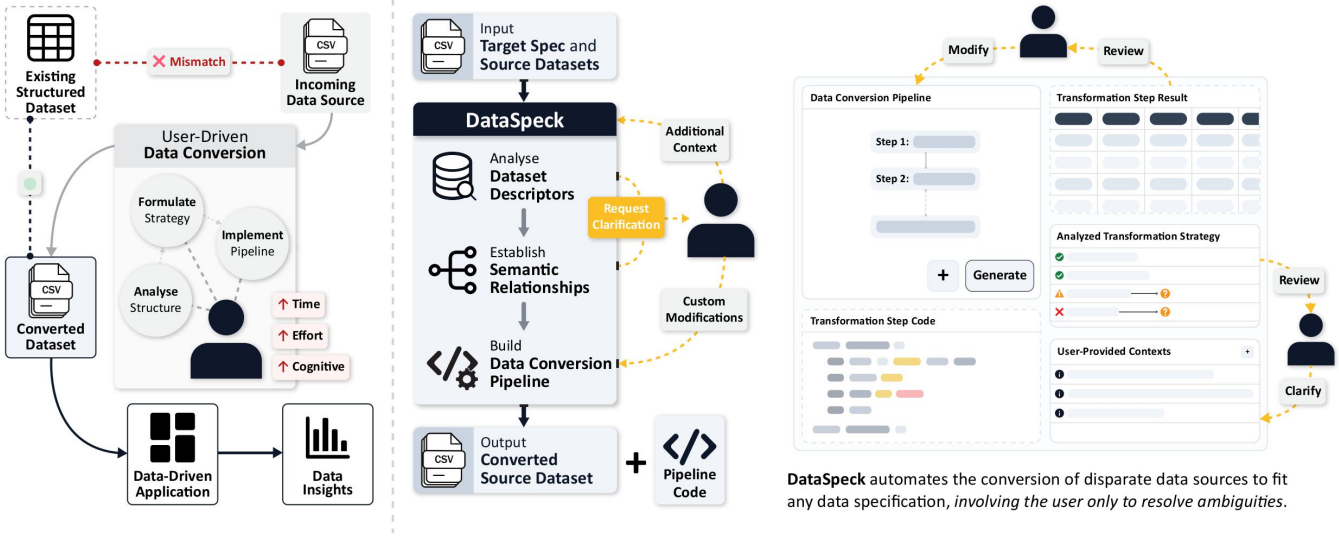

Figure 1: Integrating disparate data sources into pre-existing systems often requires manual data conversion to fit the system's required structure, which can be cognitively demanding and time-consuming. *DataSpeck* fully automates this process by analyzing dataset descriptors and inferring semantic relationships between the datasets. Instead of having users drive the conversion, our system builds the entire pipeline autonomously while keeping the user in the loop to resolve ambiguities and refine the pipeline through natural language interactions.

Abstract

In data-driven systems, integrating disparate data sources becomes challenging when incoming data doesn't conform to the system's specifications. Despite advances in automated schema matching systems, data integration tasks involving complex semantic interrelationships still require users to manually identify and define transformations between datasets, which can be cognitively demanding and time-consuming. We present DataSpeck, an end-to-end system that automates the conversion of disparate data sources to fit any pre-existing data specification. DataSpeck employs an AI-driven human-in-the-loop design, using LLMs to analyze semantic relationships and generate step-by-step transformation pipelines autonomously, while only requesting user attention to resolve semantic ambiguities. In our technical evaluation, DataSpeck successfully automated ~86% of varied data transformations while generating interpretable strategies with confidence scores and targeted clarification requests. In a user study (N=12), participants completed data conversion tasks ~53% faster with significantly reduced cognitive load using DataSpeck compared to Microsoft Excel with Copilot.

CCS Concepts

• **Human-centered computing** → **Interactive systems and tools**.

Keywords

Data Transformation, Schema Matching, Data Wrangling, Human-in-the-loop, Semantics, LLM, AI, Adaptive

ACM Reference Format:
Adil Rahman, Koichiro Niinuma, and Aakar Gupta. 2026. DataSpeck: An AI-Driven Human-in-the-Loop System for Automating Transformations in Data Conversion Workflows. In *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems (CHI '26), April 13–17, 2026, Barcelona, Spain.* ACM, New York, NY, USA, 28 pages. https://doi.org/10.1145/3772318.3791857

CHI '26, Barcelona, Spain

ACM ISBN 979-8-4007-2278-3/2026/04
https://doi.org/10.1145/3772318.3791857

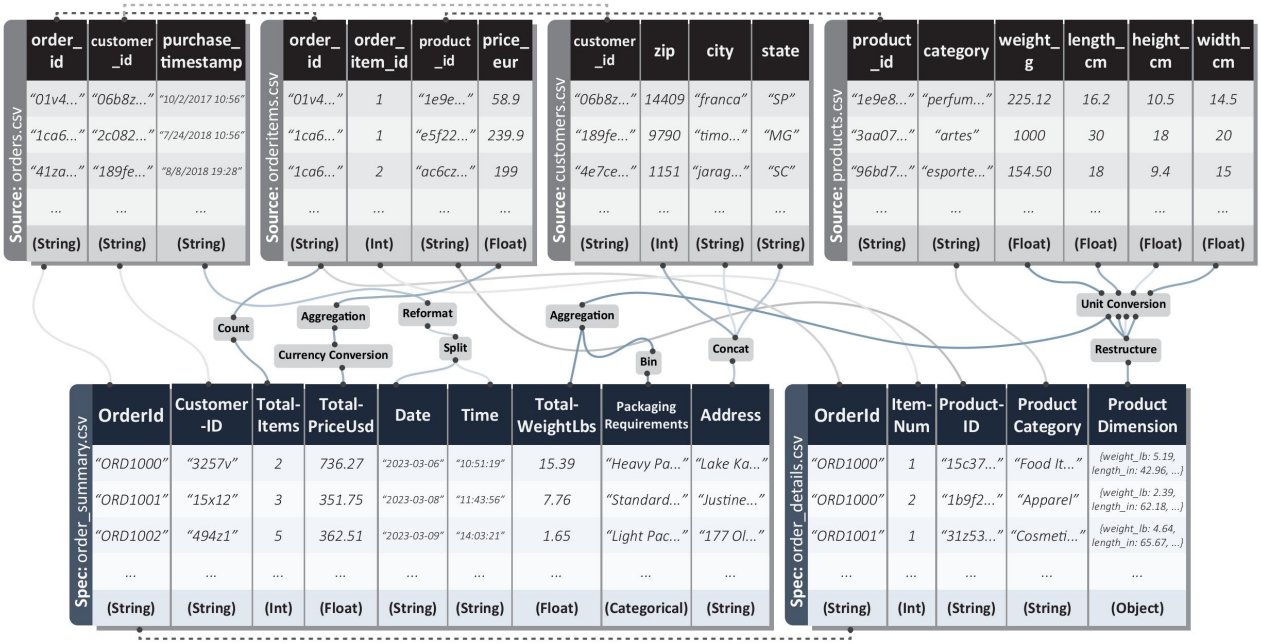

Figure 2: Datasets from the same domain but different sources can differ significantly in structure, making them interoperable with each other. This figure illustrates the various complex transformations required to convert an e-commerce dataset consisting of 4 CSV files (Kaggle [90]) into a target specification dataset consisting of 2 CSV files (derived from Kaggle [23, 111, 112])

1 Introduction

Data-driven systems are often built around specific data formats and structures optimized for their intended use cases. However, the growing variety of data sources poses a major challenge, as incoming data frequently deviates from expected formats and must be adapted before it can be utilized [59]. For example, a healthcare analytics platform designed to process standardized patient records may struggle to integrate external research datasets with differing schemas [11]. Integrating new data sources within established data architectures requires manually designing elaborate transformation pipelines which map incoming data to match the specification format [114]. This process consumes time and effort and must be repeated every time a new data source is introduced - creating significant bottlenecks in data integration workflows.

Reconciling different data formats has been a longstanding research goal spanning decades in data management and information systems, with schema matching and mapping techniques being the primary approaches to enable interoperability between heterogeneous data formats [86, 99]. While these systems excel at finding match candidates between source and target schemas, they often leave the semantic relationships and necessary transformations between matched attributes for users to define manually [6, 10, 13, 99, 127]. In data workflows, this transformation step remains a significant bottleneck, requiring detailed preparation before data can be utilized [76]. To address this, various interaction techniques seek to simplify the process by reducing manual coding requirements. Programming-by-example [8, 20, 44, 49, 63, 107] and Programming-by-demonstration [47, 65] systems have proven particularly effective in eliminating the need to manually write transformation codes and allow users to automate data formatting through demonstrated examples. Natural language interfaces [56] have further enabled users to describe transformation requirements in plain language, though they often require precise phrasing to avoid ambiguity [67, 110, 124]. Despite these advancements, users must still understand data structures, interpret relationships, and devise appropriate transformations—a process that remains cognitively demanding for complex or unfamiliar datasets. While human insight remains crucial for exploratory analysis [106], scenarios with predetermined data structures could benefit from more nuanced automation approaches that reduce unnecessary overhead when adapting diverse sources to existing specifications.

In this paper, we present *DataSpeck*, an AI-driven system designed towards converting a dataset into a prescribed specification. Unlike previous approaches, DataSpeck doesn't require users to provide examples or describe the transformation process explicitly. Instead, it leverages LLMs to analyze the semantic relationships between the new data source and the pre-existing data specification, and uses this understanding to automatically generate both transformation strategies and the corresponding data conversion scripts. However, data conversions can involve ambiguities that require additional context. To address such scenarios, DataSpeck incorporates a human-in-the-loop design, classifying the need for human input based on system confidence, and prompting users to provide additional context when necessary.

To evaluate the effectiveness of our human-in-the-loop system, we first conducted a technical evaluation to understand the boundaries of our system's automation capabilities. We tested DataSpeck

against 43 isolated transformation scenarios and 5 complex real-world data conversion scenarios. Our system was able to successfully automate ~ 86% of the data transformation operations and yielded appropriate system confidence scores and clarification requests. Our technical evaluation also highlighted transformation scenarios where the automation capabilities struggled. Using these insights, we designed a user study with 12 participants who had prior data integration experience to measure the system's impact on user performance and effort. As a baseline, we compared DataSpeck against Microsoft Excel with Copilot, which represented the counterpart human-driven, AI-in-the-loop approach where users infer the semantic relationships on their own, and then use natural language interactions to design the transformations. Participants achieved significantly higher performance and efficiency in the given data conversion tasks, completing tasks 53% faster on average using DataSpeck, and reported significantly lower levels of mental demand, effort, and frustration on the NASA-TLX scale. They found DataSpeck's ability to automatically infer transformations from source-specification pairs highly usable and valuable for their professional settings, noting that such an interaction could transform the tedious task of manually analyzing datasets and writing migration scripts into simply reviewing system-generated strategies and answering clarifications, thereby significantly reducing manual effort and cognitive load.

We summarize our contributions as follows:

(1) The design of *DataSpeck*, an end-to-end system that automates the transformation of disparate data sources into pre-existing data specifications through an AI-driven, human-in-the-loop approach.
(2) Technical findings demonstrating that DataSpeck successfully automates a comprehensive set of data transformation operations across diverse scenarios, while generating appropriate confidence scores and targeted clarification requests when human input is needed.
(3) User performance results showing that DataSpeck's AI-driven, human-in-the-loop approach enables users to complete data conversion tasks more efficiently while significantly reducing cognitive load across all NASA-TLX dimensions compared to human-driven AI-in-the-loop approaches.

2 Background & Related Work

Our work draws inspiration from prior research on reconciling heterogeneous data sources and interfaces that facilitate data transformation tasks. We now review these areas, along with recent advances in applying Large Language Models to data work and user interfaces.

2.1 Reconciling Heterogeneous Data Sources

Reconciling heterogeneous data sources has been a long-standing challenge when organizations need to integrate, analyze, or migrate data across systems. These are often laborious tasks that become increasingly complex as data volumes and variety grow. Converting datasets between different representations requires understanding how data elements relate across different schemas, a problem addressed through schema matching [99] and mapping [86] techniques. Schema matching identifies correspondences between elements of different schemas, while schema mapping generates executable transformations to convert data between formats.

Traditional approaches have relied on various heuristics: analyzing element names and descriptions [9, 16], examining data structures and constraints [68, 71], and comparing instance-level data [40, 118]. Modern systems typically combine these into hybrid matchers [30, 77, 83]. While effective for straightforward matches (e.g., "emp" to "employee"), these approaches struggle with semantic ambiguities requiring domain understanding (e.g., recognizing "buyer" as "customer" in e-commerce contexts). Recent advances have incorporated domain knowledge through knowledge bases [4, 12], semantic embeddings [51, 69], and Large Language Models [103, 105, 127], showing strong generalization even in low-data scenarios. However, despite extensive research on identifying match candidates, understanding how matched elements relate remains largely unexplored. Most approaches assume direct equality relationships (e.g., pID=productId) [13, 99] or require manual user specification for complex transformations [10, 127]. Real-world scenarios often demand sophisticated transformations—unit conversions, custom binning, string manipulations, or multi-step aggregations [14, 72]. Such complex transformations account for nearly 50% of real-world mapping cases [32], likely higher today given increasing data heterogeneity.

Consequently, schema matching remains predominantly user-driven: before practitioners can integrate new data into existing systems, they must manually examine each dataset, audit match candidates, identify semantic relationships, and explicitly specify transformations—a process that becomes prohibitive when dealing with hundreds of datasets containing thousands of schemas [10]. Even when the incoming data is correct and domain-appropriate, differences in representation and formatting require extensive manual intervention before the data becomes usable. To truly support heterogeneous data reconciliation at scale, we need intelligent systems that provide end-to-end automation throughout the transformation pipeline, strategically involving humans only when their expertise is essential rather than for routine conversions [33, 37, 75].

2.2 Facilitating Data Transformations

Specifying data transformations often requires writing complex query expressions with precise syntax. Modern schema mapping systems reduce this burden by providing visual mapping interfaces [48, 81, 84, 98, 109]. Tools like *Altova MapForce* [2] and *Talend* [96] let users draw connections between elements and configure transformations, but users must still strategize and assemble the transformation logic themselves.

Interactive tools such as Wrangler [65] and Proactive Wrangler [47] introduced programming-by-demonstration approaches, enabling users to manipulate data directly and receive transformation suggestions. These lower syntactic barriers but still require users to perform each transformation and validate the outcome. Programming-by-example systems like Foofah [63], ProgFromEx [49], and FlashFill [44] automate transformation synthesis from examples, enabling fast and intuitive wrangling. Their success has led to several inductive systems for data manipulation [26–28], though

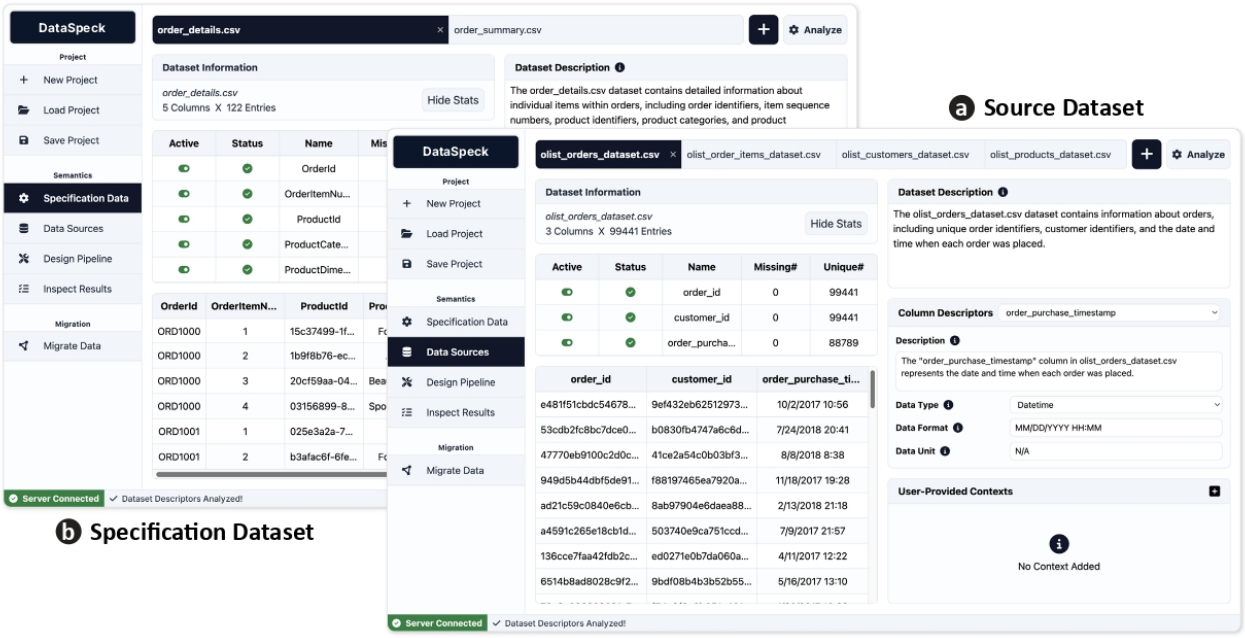

Figure 3: Dataset descriptor analysis in *DataSpeck*, illustrating the ⓐ Source and ⓑ Specification datasets from the demonstration scenario. The interface displays dataset-level summaries and column descriptors such as data type, format, and unit. The green tick next to each attribute indicates that the system is confident in its analysis and does not require user clarification.

they often struggle with multi-step or edge-case-heavy logic and depend heavily on representative examples [46]. Natural language interfaces aim to eliminate the need for examples by letting users describe transformations directly [56, 57, 116]. NL2Rigel [56] and WaitGPT [122] show how LLMs can be used for NL-driven transformation and analysis. However, these approaches rely on users articulating transformation intent precisely—a challenge for complex or ill-defined tasks—and often produce errors due to ambiguity [67, 110, 124].

While prior work has eased the process of data transformation, deciding what transformations are needed and how to compose them—remains manual. Users must still infer semantic relationships, synthesize transformation logic, and validate outputs, which can be significantly time consuming and labor intensive. Our work targets this gap by enabling end-to-end, goal-driven transformation with selective user involvement only where necessary.

2.3 LLMs in Data-Centric Tasks

Large Language Models (LLMs) have been increasingly adopted for data-centric tasks due to their ability to interpret natural language, generate transformation logic, and reason over structured inputs [55, 74]. For instance, BigBench [15] evaluates LLMs' capabilities on row-level data tasks. LLMs have also been applied to SQL generation from natural language [22, 54], simplifying access to structured data for non-technical users. However, these systems often operate at the level of individual tasks—such as writing formulas, transforming specific fields, or answering queries—rather than supporting broader, goal-driven workflows. Their outputs are also prone to errors or inconsistencies [78, 129], and conversational interfaces can place a high cognitive load on users, who must iteratively refine prompts and verify results [3, 67, 110]. In response, recent work has explored integrating LLMs into structured interfaces that constrain inputs, guide interaction, and improve reliability [31, 38, 119, 123]. These systems demonstrate the value of combining LLMs with interface design to support data work.

Recent systems have also applied mixed-initiative approaches combined with LLMs to table-related tasks. ReAcTable [128] and InReAcTable [5] target table question answering and visual data story construction, respectively. Dango [25] combines programming-by-demonstration with LLM-generated clarification questions, allowing users to articulate their intent through natural language when demonstrations are ambiguous. These systems significantly lower the barrier to performing transformations. However, demonstration and intent-driven approaches assume users already understand the data and know what transformations are needed—an assumption that breaks down in data conversion scenarios involving unfamiliar schemas or hundreds of fields requiring alignment. When the goal is simply to make incoming data compatible with a pre-existing specification, expressing intent field-by-field becomes costly and impractical. End-to-end data conversion—given a source dataset and a specification dataset—remains an open problem, as existing systems still rely on users to inspect fields, interpret semantics, and compose transformation logic [33, 37].

3 DataSpeck: System Demonstration

We now demonstrate the usage and interaction experience of *DataSpeck* through a data conversion scenario involving real-world challenges.

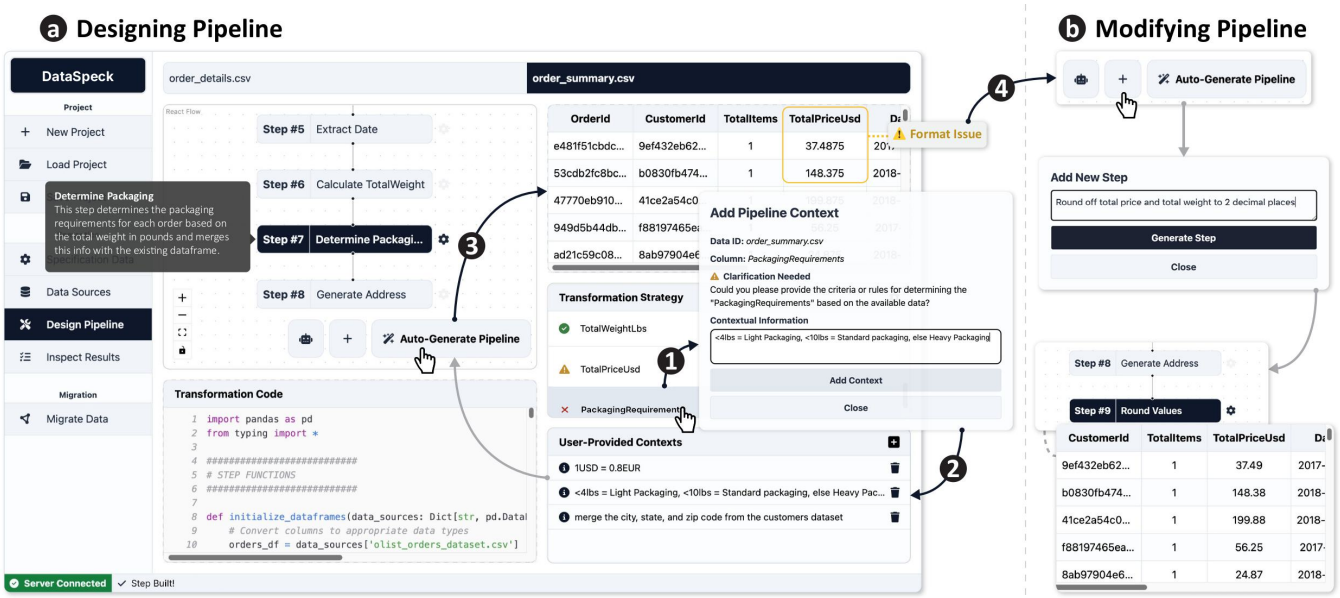

Figure 4: The pipeline designer in DataSpeck supports AI-driven pipeline generation with human-in-the-loop interaction. Once the system generates transformation strategies, the user proceeds to ① review them, ② provide any clarifications, and ③ inspect the generated pipeline by viewing step descriptions and intermediate outputs. If issues are found, the user ④ modifies the pipeline by adding new (or editing existing) steps through natural language and observing the updated results.

Consider Sana, a data analyst at a major e-commerce platform, who faces a challenging data integration task following a corporate merger. Her company's analytics pipeline expects two standardized CSV files (Specification Dataset): `order_summary.csv`, which aggregates customer and order-level metrics, and `order_details.csv`, which contains item-level product details. However, the acquired company provides its data in a very different structure that is distributed across four separate CSV files (Source Dataset): `orders.csv`, `order_items.csv`, `customers.csv`, and `products.csv`. These structural mismatches render the source data incompatible with the existing system, and converting it requires complex, multi-step transformations across related tables (Figure 2).

While Sana could turn to ETL tools like MapForce [2] or Talend [96], or even Python scripts with Pandas, these solutions still require her to first understand both data schemas and determine what transformations are needed. While these tools assist in identifying potentially related elements or offer no-code interfaces for defining mappings, the core challenge remains: she must infer how the data elements are semantically related and plan the sequence of transformation operations required to align them. This involves reasoning over complex relationships—such as calculating total order weight from nested product attributes or classifying packaging based on custom constraints—often spread across multiple linked tables, making the task both non-trivial and cognitively taxing. *DataSpeck* eliminates this burden by automating both the inference of structural relationships and the synthesis of transformation strategies.

Using *DataSpeck*, Sana can simply input the source and specification datasets, any associated meta-data, and get the transformation script and the converted dataset as output, only providing clarifications to the system when needed. As a first step, Sana loads the source and specification datasets into the DataSpeck tool. The tool analyzes both, inferring dataset descriptors such as column types, formats, and units (Figure 3) and identifying semantic relationships within each dataset. Based on this analysis, it proposes transformation strategies for generating each column in the specification dataset using the source data. Each proposed target column is explained using natural language, and any uncertainties are surfaced with clarification prompts along with system's confidence level indicators—for example, asking for a currency conversion rate or packaging rule logic (Figure 4 ①). Sana reviews these system-generated strategies and provides the required clarifications in natural language (Figure 4 ②). The system uses these additional contexts along with the dataset descriptors and inferred strategies to auto-generate a step-by-step data conversion pipeline. The pipeline can then be inspected through a visual interface showing each transformation step, a description, and intermediate outputs for review (Figure 4 ③). If Sana notices an issue—such as prices not being rounded—she can modify the pipeline by adding or editing steps in natural language (Figure 4 ④). The updated results are immediately reflected, allowing iterative refinement.

Behind the scenes, *DataSpeck* generates structured transformation code (Figure 5), organizing the pipeline as composable Python functions. The system automatically determines the structure of the pipeline, as seen in the main transformation sequence, and implements each step with reference to inferred schema linkages and foreign keys. In the given scenario, one step involves merging tables and aggregating weights with unit conversion (Figure 5 ①); others incorporate user input, such as applying a custom currency rate

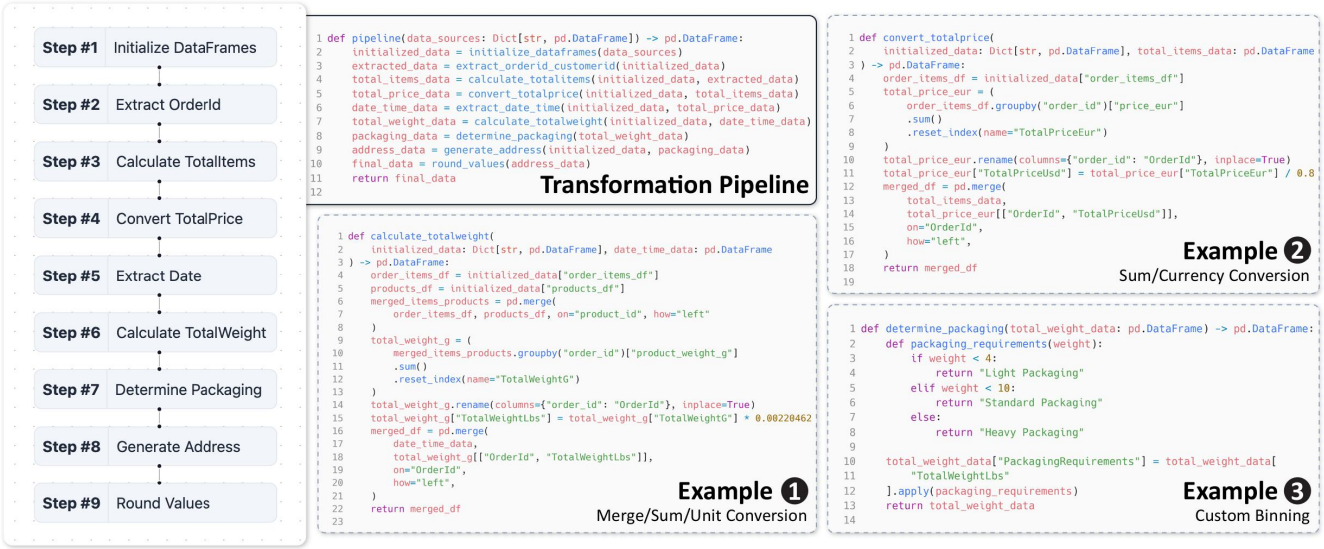

Figure 5: This figure illustrates the transformation pipeline defined by DataSpeck for the demonstration scenario, along with sample implementations of individual transformation steps. The code is autonomously generated by the system based on inferred relationships across datasets—including schema linkages and foreign keys ①. It also incorporates user input where needed: ② uses a user-provided currency conversion rate, and ③ applies user-defined logic for custom binning.

(Figure 5 ②) or user-defined binning rules for packaging classification (Figure 5 ③). Finally, when Sana is satisfied with the results, she can download the converted data along with the pipeline script for reuse or future modification.

With just 2-4 data tables with 5-10 fields, the above scenario is a toy example of the data conversion challenges in the real world where the required manual effort is order of magnitudes higher. Such challenges are common across domains where data from multiple sources contains similar information but is structured very differently. For example, a company may need to convert legacy data created before a 2015 schema overhaul; sensor data might come from multiple manufacturers, each with its own format; or a company may use public government data, such as census datasets from different countries, each following its own structural conventions. *DataSpeck* converts what is typically a manual, expertise-driven process of inferring semantic relationships and planning transformation logic into an AI-driven interaction by automatically inferring and proposing executable transformation strategies. Instead of manually designing the entire pipeline, users are kept in the loop only to resolve ambiguities and refine system-generated decisions, saving enormous manual effort and associated costs.

4 DataSpeck: System Design

We now describe the system design of DataSpeck. We first present the design rationale motivating our approach and formalize the data conversion problem it addresses. Next, we present the *Data Conversion Automation Workflow*, which describes the system's overall automation pipeline—comprising three stages—as shown in Figure 6. We then elaborate on two key components that support this workflow: the *Human-in-the-Loop* feedback mechanism, which resolves semantic ambiguities, and the *Transformation Code Generation*, which produces executable Python code for conversion. Finally, we conclude this section with a summary of implementation details.

4.1 Design Rationale

As established in the prior sections, end-to-end data conversion remains an open problem. Our goal is to address this through an AI-driven system that automates the complete workflow. However, we recognize that full automation is neither realistic nor desirable—LLMs can misinterpret semantics, hallucinate mappings, or lack domain-specific knowledge that only users possess. This necessitates keeping humans in the loop. Our overarching aims were therefore twofold: (i) **maximize automation** wherever possible, and (ii) **minimize user effort** to only what is essential.

To understand where human involvement becomes necessary, we draw on how the data conversion problem has been addressed in prior work. Existing approaches have tackled dataset understanding [99, 127] and transformation execution [63, 65] largely in isolation, while the intermediate step of establishing semantic relationships remains manual [10, 13]. Unifying these yields three stages for end-to-end conversion: **analyzing the structure and semantics** of both datasets, **establishing the relationships** between source and target attributes, and **building the transformations** that produce specification-conformant output (Figure 1). Automating these operations will inevitably encounter points of uncertainty, such as ambiguous semantics during analysis, multiple plausible mappings when establishing relationships, or insufficient context for constructing transformations. Rather than requiring continuous user oversight, our system should operate autonomously by default

and involve users selectively at these failure points, when confidence is low or when genuine ambiguity requires human judgment.

Based on this rationale, we present *DataSpeck*, an AI-driven human-in-the-loop system that automates the conversion of disparate data sources to fit pre-existing specifications. DataSpeck infers transformation strategies from the specification upfront, soliciting user input only to resolve ambiguities rather than to articulate transformation intent.

4.2 Problem Formulation

We now formalize the data conversion task and introduce the core terms and notations used throughout the system design.

Definition 1 (Dataset). A dataset D is a collection of one or more relations $D = \{R_1, R_2, \ldots, R_n\}$, where each relation R corresponds to a single CSV file containing tabular data. Each relation R consists of a set of tuples (rows) and a set of attributes (columns) $A(R) = \{a_1, a_2, \ldots, a_k\}$, where each attribute $a \in A(R)$ is identified by a name and associated with a domain of values.

We refer to the dataset that needs to be transformed as the **source dataset** D_{src}, and the target structure the data must conform to as the **specification dataset** D_{spec}. In practice, D_{src} may originate from an external or legacy system and deviate significantly in structure, naming, and formatting. The specification dataset D_{spec} reflects the desired format that is known to be compatible with the downstream system. To support open-ended, real-world data conversion scenarios, we assume that D_{src} and D_{spec} are completely disjoint, i.e., there are no shared entities across the two datasets.

Definition 2 (Transformation Function). The transformation function T is composed of a sequence of atomic operations $T = \langle t_1, t_2, \ldots, t_q \rangle$, where each step t_i represents a discrete computational or logical operation applied to the data. transformation applied to the data, such as restructuring tables, modifying formats, aggregating values, or deriving new attributes. This characterization draws from foundational work on data transformation taxonomies [45] and analyses of real-world data engineering practices [89].

Definition 3 (Dataset Conversion Problem). Given a source dataset D_{src} and a specification dataset D_{spec}, the dataset conversion problem is to construct a transformation function T such that $T(D_{\text{src}}) = D'_{\text{src}}$, where:

(1) $|D'_{\text{src}}| = |D_{\text{spec}}| = p$
(2) For each $j \in \{1, 2, \ldots, p\}$, the relation $R'_j \in D'_{\text{src}}$ has the same set of attributes as the corresponding $R_j^{\text{spec}} \in D_{\text{spec}}$, i.e., $A(R'_j) = A(R_j^{\text{spec}})$
(3) The content of D'_{src} is derived from D_{src} through data transformation operations as characterized in prior work [45, 89].

In practice, data conversion scenarios exist on a spectrum of semantic overlap between source and specification datasets. When D_{src} contains all necessary information, the transformation is fully definable. However, when gaps exist—due to ambiguous semantics, missing domain knowledge, or non-derivable attributes—user input becomes necessary. To accommodate this, we extend the formulation to $T(D_{\text{src}}, C) = D'_{\text{src}}$, where C represents optional user-provided context that supplements the source data when automated inference is insufficient.

4.3 Data Conversion Automation Workflow

The data conversion workflow serves as the core component of DataSpeck, facilitating automated conversion of source datasets to fit the structure of any given specification dataset. It accomplishes this by semantically analyzing both datasets and generating the necessary transformations using LLMs in a custom-designed workflow. The workflow consists of three stages: (a) *analyzing dataset descriptors*, where the system extracts and interprets structural and semantic characteristics of both datasets; (b) *establishing semantic relationships*, where meaningful mappings between source and target data elements are inferred using reasoning and contextual cues; and (c) *building conversion pipeline*, where the system synthesizes executable code to perform the required transformations. While the workflow operates autonomously, it is designed to incorporate user-provided contexts at every stage as needed. For clarity, we describe the automation workflow here in isolation; human-in-the-loop contributions are detailed in Section 4.4. Figure 6 provides an overview of this automation workflow.

4.3.1 Analysing Dataset Descriptors. The source and specification datasets are input to the system, optionally accompanied by supporting metadata files. We begin by constructing a compact representation of these inputs. Directly loading entire datasets into LLMs is impractical due to computational and context size constraints. To address this, we extract structural and semantic characteristics that capture the essence of the data while maintaining a manageable size. To keep the system open-ended and compatible with widely used formats, we operate on CSV files—which are flexible and broadly adopted—but that also implies they may lack inherent relational schemas. To handle this, we apply a two-step analysis process (Figure 6 ⓐ).

The **Structural Descriptor Analyzer** first processes each relation $R \in D_{src} \cup D_{spec}$ by converting it into a dataframe. For each attribute $a \in A(R)$, it extracts various structural characteristics, including the *attribute name*, *five sequential* and *ten random data samples*, *counts of missing* and *unique values*, and *mode/frequency* distributions. For attributes containing numerical values, the system also computes additional statistical metrics, including *mean*, *median*, *standard deviation*, *minimum*, and *maximum* values. Together, these characteristics constitute the structural descriptors used in the subsequent semantic analysis.

The **Semantic Descriptor Analyzer** processes the structural descriptors via a prompt-driven LLM function, which maps them into predefined prompt templates tailored to extract structured semantic information. Specifically, for each relation $R \in D_{src} \cup D_{spec}$, the analyzer generates a natural-language *description* that summarizes the overall structure and purpose of the relation. Additionally, for each attribute $a \in A(R)$, it produces four targeted semantic descriptors: an *attribute description*, *data type*, *unit*, and *formatting* convention. Extracting these descriptors requires contextual understanding that traditional deterministic methods (e.g., regular expressions, rule-based parsers, or pattern-matching techniques) typically cannot provide. While these conventional approaches can identify simple structural patterns (e.g., date formats like `DD/MM/YYYY`), they struggle with more intricate semantic patterns (such as addresses formatted as "<street>, <city>, <state><zip>") or units of measure (e.g., kilograms, grams, Celsius). LLMs are particularly adept at

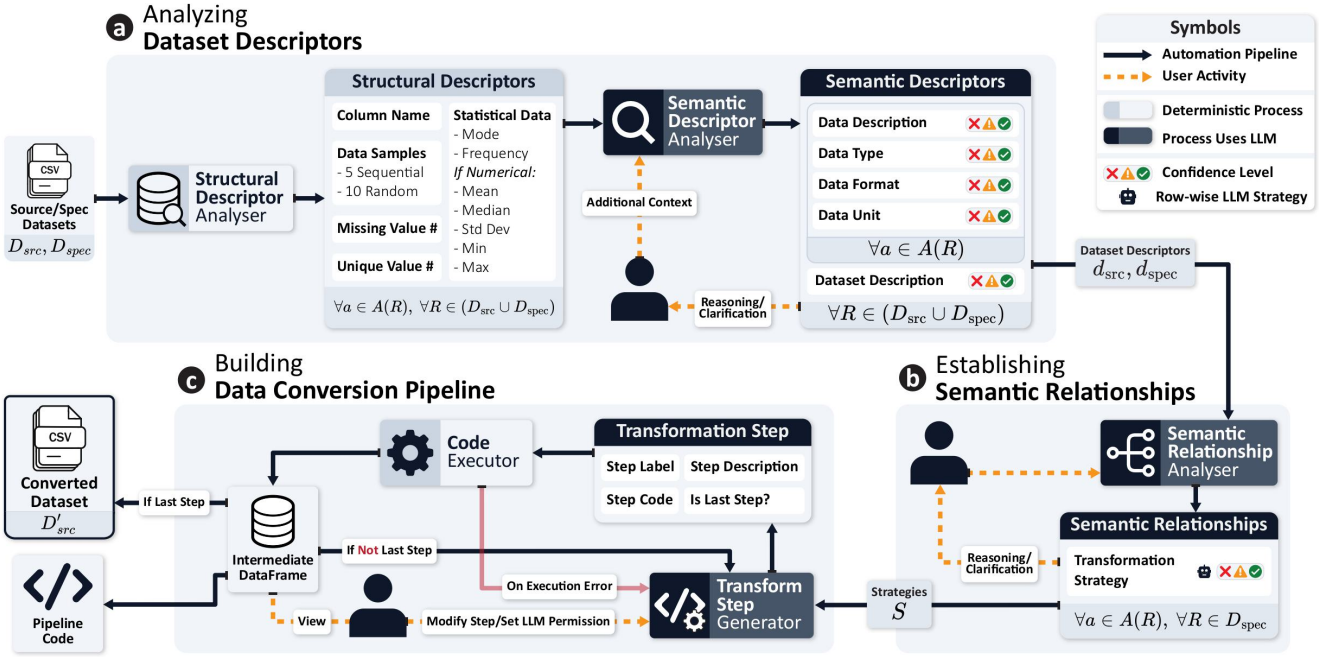

Figure 6: DataSpeck automates data conversion in three steps: ⓐ analyzing dataset descriptors to extract structural and semantic information, ⓑ establishing semantic relationships by inferring transformation strategies between attributes, and ⓒ building and executing a step-wise pipeline to convert the source data into the target specification.

capturing these nuanced semantics based on various contextual cues, enabling our system to handle complex data transformations more effectively. The combined structural and semantic descriptors form compact representations of each dataset (d_{src} and d_{spec}), which serve as inputs for identifying semantic relationships between corresponding attributes, as discussed in the next section.

4.3.2 Establishing Semantic Relationships. After generating descriptors for both the source and specification datasets, the system proceeds to determine how attributes in the specification dataset can be derived from those in the source dataset. The **Semantic Relationship Analyzer** takes as input the dataset representations d_{src} and d_{spec}, and processes them using a prompt-driven LLM function—designed similarly to the one used for semantic descriptor generation—where structured templates guide the formulation of candidate transformation strategies. For every attribute $a \in A(R)$ in each relation $R \in D_{spec}$, the analyzer formulates a natural-language transformation strategy describing how a can be synthesized using one or more attributes from D_{src}. Collectively, these natural-language, attribute-level strategies form the strategy set $\mathcal{S} = \{s(a) \mid a \in A(R),\ R \in D_{\text{spec}}\}$ (Figure 6 ⓑ). By leveraging the LLM's ability to incorporate contextual reasoning, infer latent relationships, and generalize across heterogeneous data representations, the system captures both direct mappings and more complex derived relationships that inform the subsequent conversion pipeline.

4.3.3 Building Conversion Pipeline. After establishing semantic relationships, the system translates the generated transformation strategies into executable code, forming the transformation pipeline $T = \langle t_1, t_2, \ldots, t_q \rangle$. The transformation strategies $\mathcal{S}$, along with the structured representations d_{src} and d_{spec}, are passed to the **Transformation Step Generator**, which uses LLMs to iteratively construct T, one step at a time (Figure 6 ⓒ). Each transformation step t, representing an atomic transformation operation, consists of the actual *step code* responsible for performing the data transformation, a *step label* that names the operation, and a *step description* that summarizes the intent of the step in natural language—allowing users to understand the structure and purpose of the pipeline without examining the code in detail. Additionally, each step t includes a boolean indicator *isLastStep* that determines whether it is the final step in the pipeline. This determination is made by the LLM-based step generator, which semantically analyzes the generated step code to assess whether all required attributes $A \in R$, for every relation $R \in D_{spec}$, have been generated, whether any extraneous attributes remain, and whether the resulting formats and units match those in D_{spec}. When each step t is generated, it is executed to produce an intermediate dataframe. If the step is not marked as final, the system feeds the previously generated steps back into the generator, which produces the next step in the sequence. This process continues until the final step is reached, at which point the system returns the complete transformation pipeline T and the final dataframe D'_{src}.

4.4 Involving Human in the Loop

Real-world data conversion scenarios exist on a spectrum of semantic overlap between source and specification datasets. At one end,

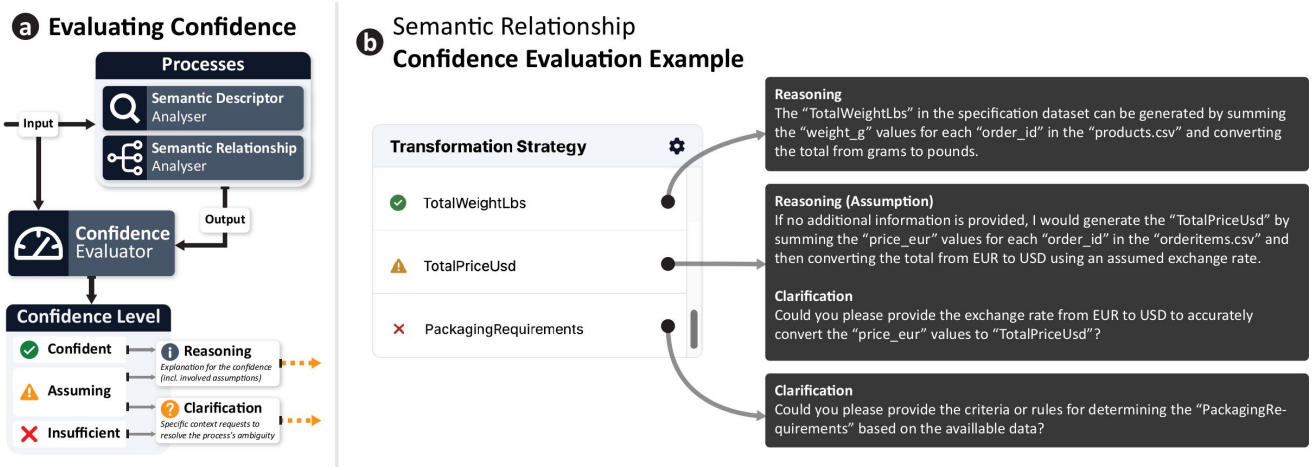

Figure 7: Semantic Inference Confidence Evaluation. DataSpeck evaluates and categorizes the confidence of its semantic descriptor and relationship analysis outputs ⓐ into *confident* ✅, *assuming* ⚠, and *insufficient* ✗ levels. ⓑ shows examples of each confidence level along with the system's reasoning/clarification prompts used to involve the user in the loop.

source data may contain all necessary information with clear correspondences to the target schema. At the other end, datasets may be largely disjoint, with minimal shared semantics or significant gaps in derivable information. Rather than assuming any particular position on this spectrum, DataSpeck is designed to accommodate the full range: it automates as much of the transformation as possible given the available data, and for aspects that cannot be reliably automated—whether due to ambiguity, missing context, or non-derivable attributes—it strategically involves the user through targeted clarification requests. This design reflects the reality that even well-matched datasets often contain edge cases requiring human judgment, while highly disjoint datasets may still share enough semantic grounding to enable partial automation.

Given the semantic variability and ambiguity inherent in real-world datasets, full, accurate automation can often be infeasible—user input is essential for producing accurate transformations. Borrowing from Zhang and Choi [126], instead of requiring constant user supervision, our system proactively identifies and highlights specific points of uncertainty, prompting the user only when necessary. To support this, both the **Semantic Descriptor Analyzer** and the **Semantic Relationship Analyzer** include a built-in confidence evaluation mechanism (Figure 7 ⓐ). When generating a semantic descriptor or transformation strategy, the system attaches a confidence label: *confident* ✅, *assuming* ⚠, or *insufficient* ✗. For confident outputs, the system includes its reasoning to support the decision. For assumed outputs, the system proposes a likely transformation but prompts the user to confirm the assumption or provide additional clarifications. For insufficient confidence, the system explicitly identifies what data or information is missing and requests user input to address the gap.

DataSpeck offers multiple interaction mechanisms to address clarifications and data insufficiency. Users can upload additional data sources if they possess files that may contain the necessary information. Alternatively, they can provide contextual clarifications through natural language input or by attaching auxiliary text-based files such as datacards (e.g., brief metadata summaries similar to those used on platforms like Kaggle). Once additional data or context is provided, users can trigger the system to recalculate confidence levels and regenerate transformation strategies incorporating the new information. Through these targeted prompts and inline visual cues (Figure 7 ⓑ), the system actively involves users in resolving ambiguity while minimizing unnecessary interactions. These user-provided contexts persist throughout the workflow (Figure 6) and are used by system components to infer more accurate descriptors, derive improved transformation strategies, and guide context-aware code generation.

In addition to surfacing semantic uncertainties, the system allows users to refine the transformation pipeline itself. Each transformation step is associated with an intermediate output, which users can inspect directly. If a step produces an unexpected result, users can revise the step by supplying additional context or by restating the desired transformation in natural language. The revised context is incorporated into the pipeline regeneration loop, enabling seamless correction without manual code editing.

4.5 Transformation Code Generation

We now outline our approach to generating transformation code for data conversion pipeline (Figure 6 ⓒ).

4.5.1 Code Structure and Step Execution. For each specification relation $R \in D_{spec}$, the system generates an independent transformation pipeline that converts relevant components of the source dataset into the desired structure of R. Each pipeline is represented as a Python `pipeline` function that takes in a dictionary of source dataset relations $R' \in D_{src}$ as Pandas dataframes and returns a transformed dataframe conforming to the specification. The system follows a modular design: each transformation step is implemented as an individual helper function, and these are invoked sequentially within the `pipeline` function to incrementally construct the

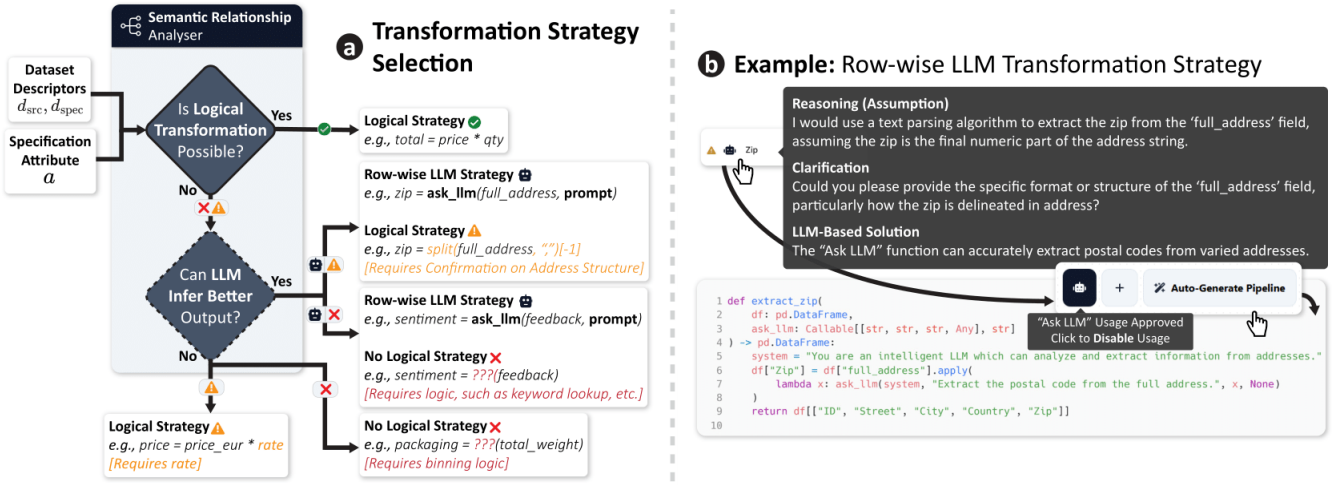

Figure 8: DataSpeck prioritizes logical strategies for generating target attributes, and falls back on row-wise LLM transformation strategies when no reliable logic is available ⓐ. In such cases, it passes relevant input fields and a generated prompt to an LLM to infer the desired output ⓑ.

output (Figure 5). The prompts used for code generation are designed to enforce this function structure, with input and output types explicitly annotated using Python's `typing` module to ensure syntactical consistency. Each step is executed immediately after generation to produce intermediate outputs and verify correctness. If an execution error occurs—e.g., due to incorrect logic or undeclared variables—the failing step and its error message are fed back to the **Transformation Step Generator**, which generates a revised version of the step. This loop continues until the code executes successfully or a timeout is reached.

4.5.2 Dynamic Step Generation. The transformation steps are constructed dynamically, with the number and granularity of steps determined contextually rather than predefined. While some conversions may align with a column-wise structure—e.g., one-to-one mappings with simple modifications—many require more semantically scoped operations that combine multiple transformations. For instance, converting a date and quantity column into a specification containing monthly averages would typically be handled in a single step involving temporal aggregation, rather than as two separate steps. The **Transformation Step Generator** iteratively produces each transformation step, evaluating after every step whether the specification has been satisfied. This goal-driven process enables the pipeline to adapt dynamically, generating contextually scoped operations that work toward the structure and semantics of the specification, rather than following a fixed attribute-wise mapping.

4.5.3 Hybrid Code Generation Strategies. When inferring transformation strategies for a specification attribute $a \in A(R)$, where $R \in D_{spec}$, the **Semantic Relationship Analyzer** first attempts to derive a logical transformation using available source data D_{src}. Such strategies can be expressed syntactically using operations such as arithmetic expressions, aggregations, or format conversions, for which executable coding scripts can be generated. However, there are situations where the required semantic information exists in the source, but cannot be easily extracted using logic alone. For example, if the source contains a crowdsourced `full_address` field and the specification requires structured outputs like `zip` or `city`, the lack of a consistent format makes reliable parsing infeasible—typically turning such cases into manual data entry tasks. However, since the necessary semantics are present in the data, an LLM can instead be prompted to extract the desired component (e.g., zip code) directly from the unstructured field [15, 89].

Our system supports such scenarios. The **Transformation Step Generator** first checks whether a logical transformation is feasible for a given attribute a. If not, it determines whether a brief clarification from the user could enable a logical strategy, or whether a more reliable result could be obtained by passing each row into an LLM along with a customized prompt (Figure 8 ⓐ). Only in the latter case is a row-wise LLM transformation generated. In such cases, the generator identifies the relevant input fields, formulates an appropriate prompt and context, and inserts a transformation step that invokes the system's `ask_llm` function. This function routes the input and prompt to a dedicated LLM used for semantic extraction and inserts the response into the transformation pipeline (Figure 8 ⓑ).

4.6 Implementation Details

Our implementation of DataSpeck consists of a user interface and a backend. The user interface is built as a web application in JavaScript using the React framework, supporting user interaction and visualization of the conversion pipeline. The backend is implemented in Python using Flask and handles all system processes, including dataset analysis, semantic inference, and pipeline generation. Socket.IO enables real-time, bi-directional communication between the frontend and backend. Both source and specification datasets, each comprising one or more CSV files, are represented as Pandas DataFrames in the backend. We use OpenAI's `gpt-4o-2024-05-13` model to support all LLM-based processes in the data conversion

workflow, including semantic descriptor inference, transformation strategy selection, confidence evaluation, and transformation step generation. Each LLM process is implemented as a modular component using Microsoft's Guidance language with constrained input/output specifications. To ensure broad applicability, prompts are designed to be general and domain-agnostic, avoiding reliance on few-shot examples. For executing row-wise LLM transformation strategies, the `ask_llm` function internally uses GPT-4o-mini as a lightweight and cost-efficient alternative.

5 Technical Evaluation

To evaluate the effectiveness of our human-in-the-loop interventions, we first need to understand the boundaries of our system's automation capabilities by identifying where it succeeds independently, where it requests user clarification, and where it fails silently. We therefore conducted a technical evaluation of DataSpeck's automation pipeline to assess its ability to infer transformations solely from source and specification dataset pairs, evaluate its handling of ambiguity through clarification requests, and identify failure modes.

5.1 Procedure

We evaluated DataSpeck using two complementary problem sets. First, we curated 43 isolated transformation tasks to systematically test the system's ability to infer individual transformation operations by solely observing two disjoint source and specification datasets. These include 11 structural transformations (e.g., column renaming, merging, splitting, pivoting) and 8 content transformations (e.g., unit conversion, format conversion, aggregation, binning), comprising 146 individual operations. For these isolated tests, we first reviewed prior work on data transformation and wrangling to compile a comprehensive set of transformation operations relevant to data conversion scenarios, synthesizing operation types from data wrangling tools including Wrangler [65], Foofah [63], Potter's Wheel [101], and established data engineering practices [45, 89] (see Appendix A.1 for full list). Based on this identified transformation taxonomy, we manually designed source-specification dataset pairs where each pair differs by exactly one transformation type, enabling us to systematically test whether the system can infer the required operation solely by observing two disjoint datasets. Second, we curated 5 compound problem sets to evaluate real-world conversion scenarios involving multiple simultaneous transformations, structural mismatches, and missing data. To ensure these tests represented real-world conditions, we sourced both source and target datasets independently from Kaggle (see Appendix Table A2), where each pair comprises completely disjoint datasets from different contributors within the same domain (e.g., two unrelated e-commerce datasets). For instance, the *Movies-1* problem pairs the Amazon Movie Ratings dataset [21] as the source with the MovieLens 100k dataset [62] as the specification—two independently contributed movie datasets that share no common entries but require the system to infer structural and semantic transformations between them.

We ran each dataset pair through DataSpeck's pipeline in a fully autonomous, zero-shot setting without human intervention, and analyzed each output against manually-defined ground truth transformations to categorize results as correct, partially correct (e.g., when the system used default values for ambiguous parameters), or unattempted, while also examining whether the system appropriately requested clarifications or failed silently when encountering gaps in derivable information. To assess whether the system's confidence classifications appropriately trigger human intervention, we manually evaluated all confidence assessments generated during the technical evaluation. For each specification column (222 in the isolated set, 45 in the real-world set), we examined the system's confidence level, generated reasoning, and clarification requests, then classified each as accurate or inaccurate. We considered "confident" classifications accurate when the transformation strategy was correct without additional input; "assuming" classifications accurate when the requested clarification was relevant and would improve the transformation; and "insufficient" classifications accurate when the system correctly identified that the column was non-derivable from available data. Two authors conducted this evaluation using a consensus-based approach: both authors first jointly established the classification criteria, then one author performed pilot labeling on a subset of cases. Through iterative discussion, the authors refined criteria boundaries and resolved ambiguous cases. One author then labeled all columns, with disagreements resolved through discussion until consensus was reached. This analysis yielded six categories (three confidence levels, each either correct or incorrect), enabling us to assess whether the system's confidence mechanism appropriately triggers interventions.

5.2 Findings

Our system yielded high automation performance on data conversion tasks, where it was able to correctly automate ~ 86% (125 out of 146) transformations in the isolated problem set, and ~ 90% (28 out of 31) transformations in the compound problem set. Our system also yielded highly accurate confidence classification of the generated transformation strategy: in the isolated set, out of 222 total specification columns, ~ 93% (175 out of 189) columns were correctly classified as confident, 100% (30 out of 30) columns as assuming, and ~ 67% (2 out of 3) columns as insufficient. In the real world set, out of 45 columns, ~ 86% (19 out of 22) columns were correctly classified as confident, ~ 86% (6 out of 7) columns as assuming, and 100% (16 out of 16) columns as insufficient. The high accurate confidence classification accuracies reflect how our system has a low chance of failing confidently or requesting human intervention unnecessarily. A detailed breakdown of the technical evaluation results is illustrated in Figure A1 (Appendix A.2).

Our evaluation revealed three distinct categories of automation performance: **(1) High-confidence automation** for transformations where clear structural or semantic relationships were directly observable between source and specification datasets—such as column renaming, format conversion, categorical remapping, and pivoting—where DataSpeck generated deterministic transformation pipelines with clearly described strategies that did not require user clarification; **(2) Partial automation requiring human guidance** where the system correctly identified transformation needs but highlighted strategies as requiring confirmation, asking targeted questions to resolve ambiguities—such as requesting binning

thresholds, currency exchange rates, or unit specifications—and appropriately defaulted to row-wise LLM transformations for complex semantic tasks like address parsing or sentiment extraction; and **(3) Accurate identification of data insufficiency** where the system correctly recognized when target columns were non-derivable from the source dataset and appropriately flagged them as requiring additional data (e.g., when the specification dataset required customer phone numbers that were absent in the provided source dataset). This capability proved particularly valuable in the real-world dataset conversion scenarios. While DataSpeck successfully decomposed complex conversions into multi-step transformation strategies, it also correctly identified all non-derivable specification columns across the five real-world dataset pairs, marking them as insufficient and generating clarifications that specified what additional data would be needed. By surfacing these data gaps upfront, such a mechanism could enable users to immediately understand data coverage limitations and identify what additional information needs to be collected or clarified—transforming what would typically be a time-consuming manual discovery process into an explicit, actionable checklist before investing effort in transformation logic.

While few, there were instances where the system failed silently. DataSpeck particularly struggled to identify transformations such as data filtering, sorting, and splitting. These detection failures primarily stemmed from the fact that such operations leave no trace in the observable data features—for example, determining that a specification dataset contains only high-value transactions requires understanding filtering criteria that cannot be deduced from examining two disjoint datasets—and in such cases, the system confidently failed because it could not recognize the gap in its inference. On the other hand, the system exhibited cautious behavior in certain ambiguous scenarios, raising an "assuming" flag to seek user confirmation even when relationships were determinable—for instance, asking whether ID formats (e.g., "ID_056" vs. "ORD056") required standardization. While rare, we also observed some cross-module inconsistencies stemming from LLM context loss and hallucination, such as correctly identifying temperature unit differences during analysis but failing to apply the conversion in the final transformation script. This was also observed in reverse in one instance, where the system initially generated an incorrect strategy during relationship analysis but accurately produced the correct transformation during code generation. Detailed technical evaluation results and complete benchmark specifications are provided in Appendix A.

6 User Evaluation

We conducted a user study to evaluate the impact of DataSpeck on user performance and effort in completing complex data conversion tasks. We aimed to understand how participants worked with our AI-driven human-in-the-loop approach to data conversion, particularly in terms of effort, usability, and overall utility—compared to a more traditional, user-driven workflow.

6.1 Baseline Comparison

No current tools to our knowledge offer end-to-end, automated support for dataset conversion based on semantic alignment of datasets. While systems like *Wrangler* [65] offer programming-by-demonstration features, they are designed primarily for efficient data preparation and transformation operations rather than end-to-end data conversion. More recent tools like *DataFormulator2* [116] support abstracted natural language statements for data creation, but focus primarily on data visualization workflows. *NL2Rigel* [57] would perhaps be closest to our use case as it enables general data transformation operations through natural language, though it limits users to NL-based interactions without UI-based transformation options. More importantly, all existing tools follow a human-driven interaction paradigm for data conversion tasks, where users first determine the necessary transformations themselves before using the tool to execute them.

We selected Microsoft Excel with *Copilot* [85] as our baseline to evaluate how users experience data conversion tasks in an unconstrained environment where they have access to multiple transformation paradigms. Excel is a mature system featuring programming-by-example (FlashFill [20]), direct manipulation, and through the Copilot extension, natural language-based transformations similar to NL2Rigel. Most importantly, all participants were already proficient with Excel, allowing us to observe the best representation of human-driven, AI-in-the-loop workflows without the confounding factor of tool unfamiliarity. This choice enables a fair contrast between our AI-driven approach, where the system infers transformations, and the traditional human-driven approach, where users must determine transformations themselves regardless of the tool's capabilities.

6.2 Study Design

Our study employed a 2x2 within-subjects design, where each participant was required to complete two data conversion tasks with each tool. The tasks mirrored real-world data conversion scenarios reflecting the complexities when dealing with diverse data sources [66, 89]. Unlike previous evaluations of data transformation tools where participants performed specific transformation operations [24, 57, 63, 65], we had end-to-end conversion tasks that required converting source dataset into the structure and format of specification dataset. Participants were provided a document describing each data conversion scenario, including the source and specification dataset structures, and any additional context regarding each of the data conversion scenarios. For the Excel baseline, participants were provided with spreadsheets pre-loaded with both specification and source datasets side-by-side formatted as tables, whereas for DataSpeck, both the datasets were pre-loaded into the system. Full details of the source and specification datasets for all four conversion scenarios are included in the Appendix.

6.2.1 Task Descriptions. Drawing from our technical evaluation results, we designed data transformation scenarios that would engage participants across the full spectrum of DataSpeck's human-in-the-loop capabilities. The study tasks thus included (1) operations that DataSpeck could fully automate with high confidence (e.g., concatenation, aggregation), enabling us to evaluate users' experience analyzing and verifying the system's transformation strategies; (2) operations that require additional context (e.g., binning, data derivation), where we could assess how users experience resolving ambiguities; and (3) operations which the system is not capable of

Table 1: Overview of the user study tasks and their corresponding operations. Operations highlighted in bold typically required additional context/manual intervention by the user (as observed in the technical evaluation).

Task Name	Operations
Tutorial Task	Column Renaming (2)
	Column Deletion (2)
	Concatenation (1)
	Reformatting (1)
	Data Derivation (2)
	Filtering (1)
Task 1: Basic Data Transformations	Column Renaming (1)
	Concatenation (1)
	Data Derivation (1)
	Data Extraction (1)
	Inner Join (1)
	Binning (1)
	Filtering (1)
Task 2: Advanced Data Restructuring	**Unfold/Unpivot (1)**
	Aggregation (2)
	Count (1)
	Lookup (1)

automating (e.g., data filtering), allowing us to understand users' manual intervention experience.

We designed the tasks to represent two different variations of data conversion complexity. The first task involved multiple transformation operations that were simple and straightforward such as data merging, concatenation, and splitting. These were relatively easy to strategize for but required managing several intermediate steps to execute, such as joining columns between two tables. The second task comprised fewer, yet more complex transformations, focusing on data restructuring and feature generation through aggregation. This task posed a greater challenge in terms of strategizing and articulating the required transformations, either manually or using natural language. Additionally, a tutorial scenario was developed to demonstrate and provide a practical tutorial for both the methods. A detailed breakdown of all operations involved in each task is illustrated in Table 1.

6.2.2 Additional Considerations. We conducted three pilot studies and iterated on the task designs to ensure users could complete them within a maximum of 35 minutes using either system. To minimize cognitive load, we limited dataset sizes: each contained approximately 6–7 columns and fewer than 100 rows.

Data conversion tasks often require users to strategize their own transformation steps, which risks introducing bias if participants perform the same task twice with different systems. To mitigate this, we created two variants of each of the two tasks described above. Each variant required the same underlying operations but used different domains and attribute names (e.g., calculating a person's age vs. time since a product was manufactured). In three pilot sessions, participants showed no signs of transfer or recognition between paired variants.

All tasks were designed to be solvable using both DataSpeck and Microsoft Excel with Copilot, relying only on logical, deterministic transformation strategies (e.g., arithmetic operations, conditional logic). To maintain fairness, we excluded operations that would require row-wise LLM transformation strategies—such as row-wise semantic interpretation—which are unavailable in Excel and would have required participants to manually fill in each row. Lastly, while participants were not provided with the explicit transformations needed to solve the conversion tasks, they were allowed to request clarifications if needed.

6.3 Study Procedure

The study was conducted in both in-person and remote settings with web-based versions of Excel and DataSpeck. At the start of the study, the experimenter explained the study scenario, obtained consent, and collected demographic information. Each session began with a tutorial on the first assigned method, after which participants completed the two data conversion tasks using that method. They were instructed to complete the tasks as quickly as possible. Upon finishing, participants filled out a post-task survey. The same process was then repeated for the second method: tutorial, tasks (with the alternate variant set), and a second survey. We counterbalanced both method order and task variant assignment across participants to mitigate order effects. A task was considered complete only after the participant had successfully performed all required transformations and the experimenter verified correctness. Participants were required to fix any errors before a task was marked complete; thus, all reported completion times include the time spent identifying and correcting mistakes. This design ensures accuracy was held constant at 100% across both conditions. Each session concluded with an exit interview, where participants reflected on their experiences with both systems and discussed the potential applicability of DataSpeck in their own workflows.

6.3.1 Data Collection. The task screen and audio were recorded throughout the study. Task completion time for each task was later computed by analyzing the screen recordings. Participants completed the NASA TLX [50] questionnaire and Likert-scale questions about system use. For DataSpeck, participants also completed the System Usability Scale (SUS) [18].

6.3.2 Measures and Analysis. Our analysis focuses on *task efficiency*, *interaction patterns*, and user-reported *task load*. Task efficiency was measured by comparing task completion times across conditions. Interaction patterns were analyzed through manual coding of screen recordings to extract user actions and phase transitions. Task load was assessed using NASA TLX responses. In addition, we evaluate the overall *system usability* of DataSpeck using participant feedback and System Usability Scale (SUS) scores. Since most data were not normally distributed, we used ANOVA tests with Aligned Rank Transform (ART) [121].

To analyze interaction patterns with DataSpeck, we manually reviewed the screen recordings and time-coded key user actions and system events, including when participants analyzed semantic relationships, added clarifications, generated pipelines, modified steps, and corrected outputs. Each session was segmented into three phases: *Task Initiation* (e.g., skimming datasets and reading the task), *Semantic Analysis* (e.g., reviewing and clarifying inferred relationships), and *Pipeline Design* (e.g., inspecting generated steps, making edits, or regenerating). These events and phases were used

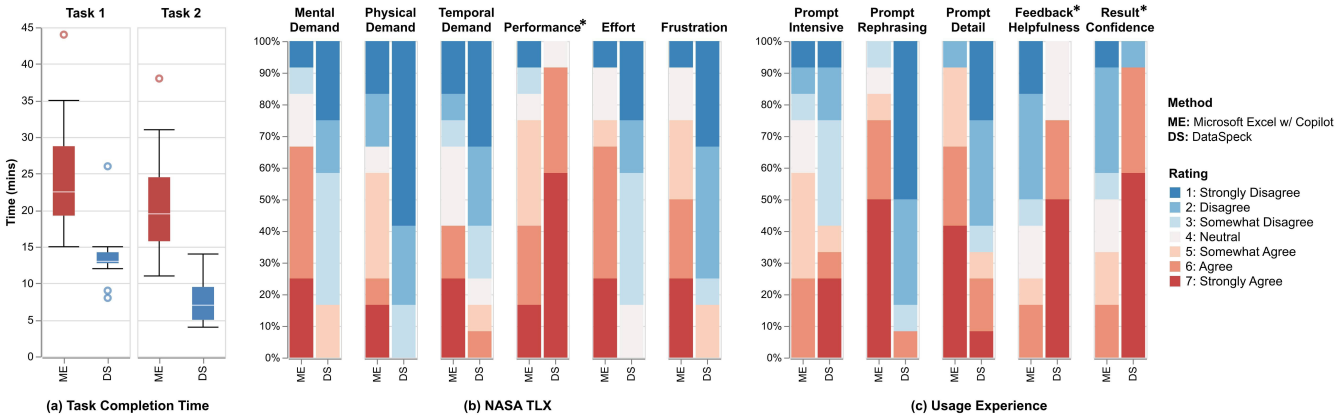

Figure 9: Study measurements comparing DataSpeck (DS) and Microsoft Excel w/ Copilot (ME): (a) Task completion time (lower is better). (b) NASA TLX dimensions and (c) usage experience metrics, where higher ratings (blue) are preferred, except for * dimensions where lower ratings (red) are preferred.

to construct normalized timelines (Figure 10), enabling comparison across tasks and participants by showing how participants distributed their effort over time. Lastly, qualitative findings were derived from thematic analysis of interview transcripts and observer notes, with themes validated across multiple participants before inclusion.

6.4 Participant Demography

We recruited 12 participants (4 Female, 8 Male, Age: 24-38 years, M=29.6, SD=4.1) who had prior experience working with Excel and often performed data conversion tasks. The group included 3 masters students, 8 PhD students, and 1 industry professional from computer science, engineering, statistics, and design backgrounds. Some student participants also had prior experience working several years in the industry dealing with similar data conversion tasks. On a scale of 1 (Novice) to 10 (Expert), participants self-reported above average expertise in using Excel (M=6.25, SD=1.09, Range=5-8) and performing data transformations (M=6.5, SD=1.5, Range=3-9). 9 users participated in-person, while 3 participated remotely via Zoom. Each participant was compensated 40 USD for two hours of their time.

7 Evaluation Results

We now present findings on task performance, workload, and overall interaction experience during data conversion tasks, comparing AI-driven automation against user-driven workflows from our user evaluation.

7.1 Task Performance, Workload, and Interaction Metrics

7.1.1 Task Performance. As illustrated in Figure 9a, participants completed the data conversion tasks ~53% (12.04 minutes) faster on average using DataSpeck compared to Microsoft Excel. For Task 1, participants were ~44% (10.75 minutes) faster with DataSpeck (M = 13.75, SD = 4.23) than with Excel (M = 24.50, SD = 8.32). For Task 2, participants were ~63% (13.33 minutes) faster with DataSpeck (M = 7.67, SD = 3.04) than with Excel (M = 21.00, SD = 7.74). These differences were statistically significant for both Task 1 ($F(1, 11) = 26.07$, $p < 0.001$) and Task 2 ($F(1, 11) = 34.79$, $p < 0.001$). Importantly, these time differences reflect total task duration including any error identification and correction—tasks were marked complete only after experimenter verification, ensuring 100% accuracy across both conditions. Since the LLM-component in both systems employed GPT-4 with comparable response times, the observed differences can be attributed to their distinct operational characteristics. Excel's completion time depended primarily on participants' ability to strategize and effectively prompt Copilot or manually perform transformations. In contrast, DataSpeck's completion time was determined by the number of transformation steps required, as the system generates transformations incrementally, with each step adding to the overall duration. Furthermore, while subsequent steps were being generated, participants could review previously generated results, effectively parallelizing the validation process with the generation of new transformations. While both systems generally performed reliably, API issues encountered during trials with two participants resulted in slower response speeds and degraded output quality, requiring multiple iterations and consequently increasing task completion times for both systems.

7.1.2 Task Load. The NASA TLX results, as illustrated in Figure 9b, demonstrated significant reductions in user effort and workload across all dimensions when using DataSpeck compared to Excel. Participants reported significantly lower effort ($F(1, 11) = 42.50$, $p < 0.001$) and mental demand ($F(1, 11) = 43.70$, $p < 0.001$) with DataSpeck, likely because the system automated much of the strategic planning required for data transformations, thereby reducing cognitive load. Participants also experienced significantly lower frustration ($F(1, 11) = 33.70$, $p < 0.001$) and physical demand ($F(1, 11) = 32.12$, $p < 0.001$) with DataSpeck, which can be attributed to the system's targeted questions that eliminated the need for repetitive or detailed prompting. Additionally, participants reported significantly higher performance ($F(1, 11) = 23.44$, $p < 0.001$) and lower

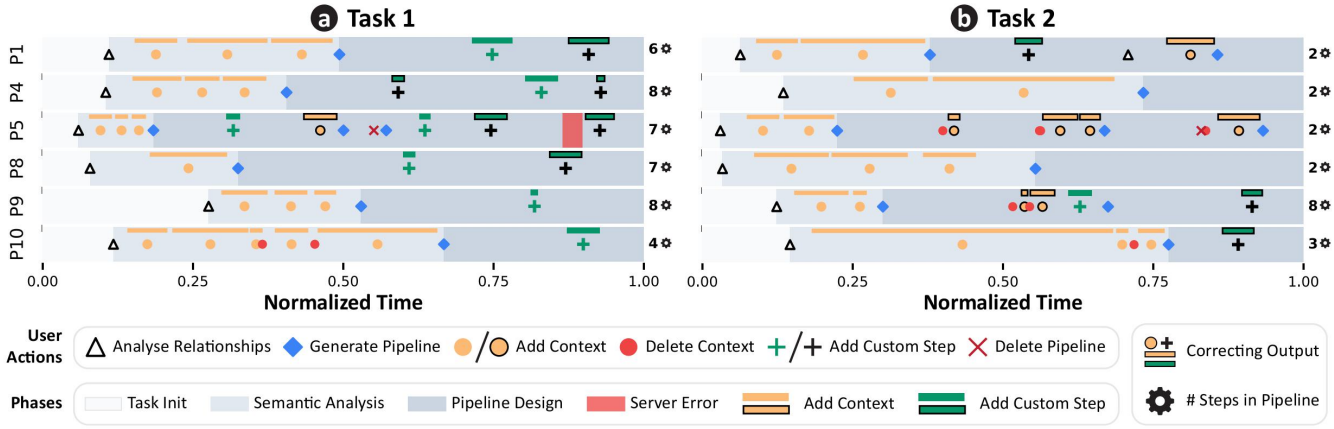

Figure 10: A sample of normalized participant interaction timelines across both user study tasks using DataSpeck.

temporal demand ($F(1, 11) = 8.39$, $p < 0.05$) with DataSpeck, consistent with the faster task completion times observed.

7.1.3 Prompt Efficiency. These workload differences can be further understood through participants' ratings of their interactions with the AI agents in both systems (Figure 9c). We assessed various aspects including prompting requirements, helpfulness of AI-generated feedback, and confidence in results. While participants perceived no significant difference in overall prompt intensity between the two methods, they reported significantly less need to rephrase prompts ($F(1, 11) = 57.76$, $p < 0.001$) or provide excessive detail ($F(1, 11) = 22.21$, $p < 0.01$) when using DataSpeck compared to Excel's Copilot. Participants also found DataSpeck's feedback significantly more helpful for guiding prompt refinement ($F(1, 11) = 17.98$, $p < 0.01$) and expressed higher confidence in its results ($F(1, 11) = 27.92$, $p < 0.001$). These outcomes align with DataSpeck's design—by pre-establishing the conversion goals and strategies, the system requires only targeted clarifications from users. This corresponds with the lower effort and physical demand observed in the NASA TLX results, indicating that DataSpeck provides more streamlined interactions and clearer feedback, ultimately enhancing the user experience for data conversion tasks.

7.1.4 System Usability. Participants had a positive experience using our system, with participants attributing DataSpeck as *easy to use* (P2, P4, P5, P8, P10), *helpful* (P3, P5, P10), and *intuitive* (P7, P12). Analyzing the SUS questionnaire responses, we found that DataSpeck achieved an average SUS score of 84.8 (Median = 85, SD = 10.2), indicating excellent usability [7].

7.2 User Strategies and Experiences

Participants overwhelmingly appreciated DataSpeck's automation capabilities, with P6 describing it as a *"plug and play setup"* where the system handles *"the heavy lifting"* of transformation design. This automation meant users didn't need to be *"invested all the time"* (P11), substantially reducing their effort. We present key themes that emerged from how participants adapted to and leveraged this automated workflow.

7.2.1 Automated Strategies Guided Users to Understand Data Progressively. The contrasting human-AI collaboration models in Excel and DataSpeck led to fundamentally different approaches to data understanding. In Excel, participants followed a traditional workflow: they first spent considerable time examining both datasets to understand their structure and content, mentally planning the required transformations, before implementing them through Copilot prompts, formulas, or manual edits. This approach required participants to develop a comprehensive mental model of the data before beginning any transformation work.

With DataSpeck, participants spent minimal time inspecting raw data upfront, as shown by the brief *Task Init* times in Figure 10. Instead, they immediately initiated the semantic analysis and learned about their data through the system's transformation strategy. Rather than building their own mental model beforehand, participants discovered data relationships through DataSpeck's descriptions and proposed transformations. They reviewed each generated step, validated intermediate outputs, and provided clarifications when prompted. Through this iterative process—reviewing strategies, validating outputs, and refining transformations—participants developed the understanding needed to complete the conversion. This allowed them to gain data insights progressively, learning what they needed precisely when they needed it.

7.2.2 Lack of Intent Understanding Can Make AI Assistance Counterproductive. There is a natural tendency for people to gravitate toward natural language interactions [88], which was reflected in our study. Participants initially attempted to use Excel's Copilot for all transformations, but its effectiveness varied dramatically by task complexity. While Copilot handled Task 1's straightforward transformations well, Task 2's complex requirements proved difficult to express through prompts—only two participants succeeded using Copilot alone. P1 noted: *"It was doing well for easy, one sentence commands, like getting min/max/avg from a column."* Similarly, P6 found Copilot *"helpful in reformatting columns and following basic prompts"* for Task 1, but for Task 2 felt it *"was not able to understand or solve some of the prompts... may require prompt engineering."*

After multiple failed attempts to articulate complex transformations, participants shifted to manual work, *"It could've been my*

prompting issue but it was more demanding for me to think of how to ask and prompt rather than me just doing it myself" (P4), while using Copilot only for formula assistance, *"I found it easier to do things by myself instead of relying on Copilot. However, for transformations where I didn't know the formula immediately, Copilot helped me get started"* (P12). This highlights a critical limitation: general-purpose AI assistants without domain-specific understanding can create additional burden. In our use case, not only did the users have to devise transformation strategies but also translate them into prompts the AI can understand, which was often harder than performing the task manually.

In contrast, DataSpeck uses the same underlying AI but aligns it to the specific task of data conversion. While it also requires user input through clarifications, these are targeted questions about specific ambiguities rather than open-ended prompt engineering. Users could describe requirements simply because the system already understood the conversion context. The nature of clarifications also differed between tasks for DataSpeck. Task 1 generated 3-4 clarification prompts on average for custom logic like binning rules or reference dates (Table 1), plus required manual filtering that the system couldn't infer. Task 2 generated only 1-2 prompts, typically about attribute mapping confirmation. Though Task 2's clarifications were simpler, participants spent more time reviewing data to understand the system's questions, consequently allowing users to learn about the data through the process. These findings suggest that AI tools for specialized tasks benefit from domain-specific framing rather than general-purpose interfaces. By constraining the problem space and understanding user intent upfront, systems can reduce the communication burden and make AI assistance genuinely helpful rather than an additional obstacle

7.2.3 AI's Intervention Requests Built User Confidence. While autonomous AI systems that generate solutions without user input can erode trust [117], we found that DataSpeck's targeted intervention requests had the opposite effect—they increased user confidence in the system's outputs. Participants valued the system's apparent ability to recognize its own limitations. P6 noted that *"the system knows when to prompt you for additional context, which increases confidence in the system."* When DataSpeck lacked sufficient information, it made explicit assumptions while informing users about them, rather than silently guessing. This transparency resonated with participants: *"I like that the system asks about something it's not sure about, making it feel more reliable, unlike how AI tools usually make up results anyway without informing"* (P2).

However, this trust mechanism can potentially have an unintended side effect: in our study we observed some participants beginning to use confidence indicators as a filtering heuristic. P2 admitted focusing primarily on warnings and errors while implicitly *"trusting results marked with green checks"*. This selective attention suggests a double-edged dynamic—while intervention requests build trust by demonstrating the system's self-awareness, they may also lead users to over-rely on the system's confidence assessments. This highlights the need for balanced design: systems should leverage uncertainty to build trust while ensuring users maintain appropriate oversight across all transformations, not just those flagged as uncertain.

7.2.4 Automated-First versus Automate-Right Approaches. Participants adopted two distinct strategies when using DataSpeck. Most took an *automate-right* approach—carefully reviewing transformation strategies and providing detailed clarifications upfront, resulting in pipelines that required minimal modification. Others followed an *automate-first* philosophy, providing brief inputs and treating generation as exploratory, adjusting outputs as they appeared (e.g., P5). These approaches produced different outputs: general inputs yielded shorter, abstract pipelines (P10), while detailed inputs generated longer, explicit step sequences (P9). Interestingly, participants who preferred the automate-first approach expressed desire for even more aggressive automation. P9 suggested the system could *"take a try right without asking"* and first attempt transformations independently: *"make those assumptions then ask me that okay I made these assumptions, now you let me know if I did right or wrong."*

This variation reveals an important design consideration: users have different preferences for the automation-control tradeoff. Some prioritize getting transformations right initially through careful specification, while others prefer rapid iteration with post-hoc correction. Both groups valued maintaining oversight, but differed in when they wanted to exercise it—before or after generation. Future systems could accommodate both styles by offering modes that balance upfront clarification against post-generation validation, allowing users to choose their preferred workflow while maintaining the trust that comes from human oversight.

8 Discussion

Our study demonstrates the potential of AI-driven, human-in-the-loop systems to drive end-to-end complex data conversion tasks. DataSpeck enables participants to review system-generated transformation strategies, respond to targeted clarification prompts, and iteratively refine the conversion process—without needing to plan or implement every step manually. Rather than front-loading their understanding of the data, participants developed insight through interaction with the system's reasoning. This approach led to faster task completion, reduced effort, and stronger user confidence, showing how automation can be paired with lightweight user input to create more efficient and transparent workflows. In this section, we discuss how specification-driven transformations can address prevalent data conversion challenges, examine the relationship between automation and data comprehension, explore the dynamics of trust in AI-generated transformations, and identify opportunities for future enhancements.

8.1 Enabling Specification-Driven Conversions

Data conversion scenarios where incoming data must be transformed to match pre-existing specifications are pervasive across data-intensive domains. In our study, every participant identified this as a pressing bottleneck in their regular workflows, with the problem manifesting across diverse contexts: when collaborating with external laboratories (P8), handling crowdsourced or consumer data (P4, P11), processing data from diverse sensors or hardware manufacturers (P12), or applying previous analyses to new datasets (P2, P6). Despite the prevalence of these scenarios, transformation paradigms like programming-by-demonstration or programming-by-example still require users to manually comprehend diverse data

formats (P4, P8) and create one-off scripts (P6, P12) to bridge the gap between incoming data and target specifications. Specification-driven transformations can be particularly useful for such instances where users provide the desired output schema and additional context to guide the transformation process, rather than demonstrating the transformation through examples [82].

This specification-driven paradigm enables diverse use cases across data workflows. In our study, participants proposed several innovative applications for this interaction paradigm. Some suggested creating various mock specifications with 4-5 rows of placeholder data to gain initial insights for multiple data analyses tasks. P6, who frequently replicated experimental setups from ML literature, noted that they could create specifications based on published descriptions to automatically transform experimental data. P2 and P4, who frequently analyzed open-ended responses from user studies and customer feedback, proposed creating mock specifications with thematic codes as placeholders, and using DataSpeck's row-wise LLM transformation function to guide their preliminary analysis. For ETL pipelines, P9 and P12 proposed that specification-driven systems could sit at the receiving end to harmonize data from various sources – such as standardizing sensor data from different manufacturers for visualization dashboards – while issuing notifications for users clarifications in case of ambiguities. This paradigm could enhance existing data tools: spreadsheet applications like Excel could offer specification-based transformation options where users provide target schemas, database systems could allow schema-to-schema mappings without explicit transformation rules, and analysis platforms could automatically adapt incoming data to match their expected formats. By integrating specification-driven capabilities, these tools could reduce the friction of data preparation while maintaining user control through targeted interactions.

8.2 Balancing Automation with Data Understanding

DataSpeck's automation removes the need for users to manually inspect raw data and design transformation scripts. This raises a critical concern about whether such automation can potentially diminish users' understanding of their data. While deep understanding of data is a crucial requirement for exploratory analysis, for scenarios where incoming data must be transformed to fit existing specifications, users may not always need intimate knowledge during data preparation. By automating these conversions, users can quickly get data into familiar formats where patterns become visible through standard analysis workflows, allowing understanding to develop more effectively while focusing on interpretation rather than transformation mechanics. This is especially valuable in industrial settings where data sources often contain hundreds of tables with thousands of fields, making manual transformation overwhelming [92]. Moreover, our study showed that participants gained data understanding despite the automation process – by reviewing transformation strategies and resolving clarification requests, they developed insights into the incoming data's structure and semantics. This can suggest that our system's automation doesn't entirely eliminate data understanding but instead provides an alternative path to it, enabling users to understand the data through guided interaction rather than exhaustive manual work, where users engage with data at a higher conceptual level while the system handles implementation details.

8.3 Balancing Automation and Trust

Prior studies have shown that some users tend to place high trust in AI-generated outputs, potentially overlooking critical details [102]. While AI can significantly accelerate data transformation tasks, there is an inherent risk of the AI making mistakes. When discussing trust in AI-generated transformations, participants (P3, P4, P6, P11, P12) compared the experience to working with a human colleague, noting that both require similar scrutiny. This was evident in the study, as participants employed various strategies to validate conversion results.

In our system's human-in-the-loop design, each analysis is accompanied by AI-generated confidence indicators [43]. Our technical evaluation demonstrated that these confidence classifications are well-calibrated, with high accuracy across all three confidence levels (Section 5.2); however, the incorrect confident classifications that remain (~7% in isolated tests, ~14% in real-world scenarios) still represent non-trivial deployment risk, as these cases provide users no signal to scrutinize potentially flawed outputs. Production deployments should therefore incorporate complementary validation processes, such as sample-based audits or automated sanity checks on output distributions, to catch silent failures. Beyond these technical limitations, confidence indicators can also create unintended trust dynamics at the user level. Our user study revealed that participants tended to *"trust the checkmarks"* (Section 7.2.3), consistent with prior findings that users often over-rely on AI-generated outputs [19]. More fundamentally, LLMs may generate plausible but incorrect mappings—for instance, confidently mapping semantically similar but functionally distinct fields. This highlights the need for more robust validation mechanisms that can provide helpful guidance while encouraging appropriate skepticism. Research suggests that reducing verification costs through targeted explanations can further mitigate overreliance [113]. A possible direction was suggested by P9: *"[In some cases] the data looks right but is not right, so I would want the AI to also give me unit testing measures that can verify [the data] with some wild tests... If [the system] actually does numerical inspection, it would give me a lot more confidence in seeing that, okay, I checked the numbers, and they're good."* Future iterations could incorporate automated test case generation and differential analysis to make validation more efficient.

Beyond individual transformation errors, in multi-step pipelines, early transformation errors can cascade through subsequent steps, compounding inaccuracies. Research on data provenance tracking [41, 42] has shown that maintaining lineage information enables tracing results back to their sources. Integrating similar capabilities into DataSpeck could help users diagnose and correct errors that propagate through generated pipelines, further supporting appropriate trust calibration.

8.4 Broadening Access to Domain-Specific Data Transformation

Data transformation challenges can disproportionately burden resource constrained organizations such as NGOs, non-profits, and community groups, which often lack dedicated engineering support

or commercial ETL tools [115]. Yet these organizations routinely integrate donor records, partner data, or public datasets, making accessible transformation tools especially valuable. Beyond resource constraints, analysts frequently face interpretive challenges when working with unfamiliar, multi-hundred-column datasets produced by other teams; prior work shows that data interpretation is deeply contextual and varies across domain backgrounds, roles, and cultures [87, 91, 94]. By surfacing inferred semantic relationships and requesting targeted clarifications, DataSpeck could help analysts uncover unexpected inconsistencies or edge cases, supporting "data discovery" through structured human–AI dialogue [87]. Human-induced data-wrangling errors—including incorrect joins or semantic misalignments—also remain common [35] and may be amplified under limited time or verification resources; DataSpeck's transparent, stepwise pipeline visualization could help surface such issues. Looking forward, adapting the system to diverse organizational contexts presents concrete opportunities: configuring domain-specific knowledge through RAG pipelines [39, 73], integrating institutional semantics via emerging standards like MCP, and leveraging multilingual LLM capabilities to bridge language mismatches in global collaborations. These directions outline how the broader HCI community might investigate domain-specific needs and extend DataSpeck toward more equitable data transformation support.

8.5 Toward Collaborative Data Conversion

Real-world data integration is inherently collaborative: knowledge about large, multi-table datasets is distributed across teams, with different members understanding different subsets [79, 125]. This distribution creates opportunities for collaborative human-in-the-loop transformation. When DataSpeck surfaces clarification requests, domain experts across sales, engineering, or finance could address ambiguities tied to their expertise. Prior work shows that supporting division of labor improves analytical efficiency and accuracy [52, 58], suggesting multi-user support as a natural next step. Future versions could enable delegated clarifications, shared visibility with scoped edit permissions, and audit trails—patterns aligned with collaborative analysis tools [79]. These possibilities raise broader questions for the community: how to reconcile conflicting interpretations, how teams coordinate distributed clarifications, and how asynchronous versus synchronous workflows differ. Addressing these questions links DataSpeck to ongoing work on cooperative data practices and collaborative AI systems [3] and highlights a concrete direction for future research.

8.6 Preserving Data Privacy

Our studies used off-the-shelf third-party LLM APIs (GPT-4), which may be unsuitable for sensitive data subject to HIPAA, GDPR, or organizational policies [36, 108]. However, DataSpeck's architecture offers flexibility in addressing these concerns. As noted in Section 8.1, participants proposed creating mock specifications with placeholder data to guide transformations—since our system operates primarily on schema-level semantics rather than raw data values, users can provide representative schemas without exposing sensitive records. Additionally, our system can operate with locally-deployed models; recent work on tabular-specific language models [34, 53] suggests that smaller, fine-tuned models could improve reliability while reducing computational costs. Such domain-specific models could operate within organizational boundaries while maintaining the semantic reasoning capabilities central to DataSpeck's approach, though tradeoffs exist between privacy preservation and access to broad world knowledge.

8.7 Handling Schema Brittleness

Our evaluation used standard Kaggle datasets, but enterprise systems may employ idiosyncratic naming conventions or domain-specific semantics that challenge LLM inference. DataSpeck's design partially addresses this through support for auxiliary datacards and active clarification requests. Additionally, our system operates on a columnar level, assuming each column represents a singular concept—yet some datasets use key-value structures where one column specifies a data type and another holds corresponding values, effectively encoding different concepts across rows rather than columns. Such structural patterns fall outside our current transformation model. Future work could address these challenges by leveraging structured data formats with richer metadata (e.g., relational databases with explicit constraints) to improve inference accuracy, or by incorporating a preprocessing stage that restructures non-columnar data into standardized tabular formats before transformation.

9 Conclusion

Data conversions often require significant manual effort to align diverse data sources to specific structural requirements. Traditional methods do not directly solve for this problem and using existing tools requires extensive manual efforts; in contrast, DataSpeck employs an end-to end AI-driven human-in-the-loop approach that understands the data, strategizes the conversion pipeline, and executes the transformation steps in a highly transparent manner, employing a tiered confidence-based human intervention mechanism when it may be needed. A technical evaluation shows that DataSpeck was able to handle a large number of transformation tasks without human intervention, automate a large part of the pipeline for real-world end-to-end conversions, and successfully surface the need for human intervention for the rest. Our user study demonstrates that DataSpeck's AI-driven human-in-the-loop approach is significantly more efficient in terms of time and effort compared to a familiar human-driven AI-assisted method. While fully human-driven methods offer complete control, they are often time-consuming and cognitively demanding. In contrast, a human-in-the-loop approach, where the AI automates the majority of transformations and requests user input for clarification or uncertainties, demonstrated potential to significantly enhance efficiency and reduce cognitive overheads for such data tasks.

References

[1] Bogdan Alexe, Wang-Chiew Tan, and Yannis Velegrakis. 2008. STBenchmark: towards a benchmark for mapping systems. *Proceedings of the VLDB Endowment* 1, 1 (Aug. 2008), 230–244. doi:10.14778/1453856.1453886

[2] Altova. 2025. Data Mapping Tools: MapForce. https://www.altova.com/mapforce. Accessed: 2026-01-24.

[3] Saleema Amershi, Dan Weld, Mihaela Vorvoreanu, Adam Fourney, Besmira Nushi, Penny Collisson, Jina Suh, Shamsi Iqbal, Paul N. Bennett, Kori Inkpen, Jaime Teevan, Ruth Kikin-Gil, and Eric Horvitz. 2019. Guidelines for Human-AI Interaction. In *Proceedings of the 2019 CHI Conference on Human Factors in*

Computing Systems (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–13. doi:10.1145/3290605.3300233

[4] Sarawat Anam, Yang Sok Kim, Byeong Ho Kang, and Qing Liu. 2016. Adapting a knowledge-based schema matching system for ontology mapping. In *Proceedings of the Australasian Computer Science Week Multiconference* (Canberra, Australia) *(ACSW '16)*. Association for Computing Machinery, New York, NY, USA, Article 27, 10 pages. doi:10.1145/2843043.2843048

[5] Gerile Aodeng, Guozheng Li, Yunshan Feng, Qiyang Chen, Yu Zhang, and Chi Harold Liu. 2025. InReAcTable: LLM-powered Interactive Visual Data Story Construction from Tabular Data. In *Proceedings of the 38th Annual ACM Symposium on User Interface Software and Technology (UIST '25)*. Association for Computing Machinery, New York, NY, USA, Article 139, 16 pages. doi:10.1145/3746059.3747719

[6] David Aumueller, Hong-Hai Do, Sabine Massmann, and Erhard Rahm. 2005. Schema and ontology matching with COMA++. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data* (Baltimore, Maryland) *(SIGMOD '05)*. Association for Computing Machinery, New York, NY, USA, 906–908. doi:10.1145/1066157.1066283

[7] Aaron Bangor, Philip Kortum, and James Miller. 2009. Determining what individual SUS scores mean: adding an adjective rating scale. *J. Usability Studies* 4, 3 (may 2009), 114–123.

[8] Daniel W. Barowy, Sumit Gulwani, Ted Hart, and Benjamin Zorn. 2015. FlashRelate: extracting relational data from semi-structured spreadsheets using examples. In *Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation* (Portland, OR, USA) *(PLDI '15)*. Association for Computing Machinery, New York, NY, USA, 218–228. doi:10.1145/2737924.2737952

[9] Glenn B. Bell and Anil Sethi. 2001. Matching records in a national medical patient index. *Commun. ACM* 44, 9 (Sept. 2001), 83–88. doi:10.1145/383694.383711

[10] Zohra Bellahsene, Angela Bonifati, Fabien Duchateau, and Yannis Velegrakis. 2011. On Evaluating Schema Matching and Mapping. In *Schema Matching and Mapping*, Zohra Bellahsene, Angela Bonifati, and Erhard Rahm (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 253–291. doi:10.1007/978-3-642-16518-4_9

[11] Ashwin Belle, Raghuram Thiagarajan, S. M. Reza Soroushmehr, Fatemeh Navidi, Daniel A. Beard, and Kayvan Najarian. 2015. Big Data Analytics in Healthcare. *BioMed Research International* 2015, 1 (2015), 370194. doi:10.1155/2015/370194

[12] Jacob Berlin and Amihai Motro. 2002. Database Schema Matching Using Machine Learning with Feature Selection. In *Proceedings of the 14th International Conference on Advanced Information Systems Engineering (CAiSE '02)*. Springer-Verlag, Berlin, Heidelberg, 452–466.

[13] Philip A. Bernstein, Jayant Madhavan, and Erhard Rahm. 2011. Generic schema matching, ten years later. *Proceedings of the VLDB Endowment* 4, 11 (Aug. 2011), 695–701. doi:10.14778/3402707.3402710

[14] Stefan Biffl, Olga Kovalenko, Arndt Lüder, Nicole Schmidt, and Ronald Rosendahl. 2014. Semantic mapping support in AutomationML. In *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*. IEEE, Barcelona, Spain, 1–4. doi:10.1109/ETFA.2014.7005276

[15] BIG-bench Authors. 2023. Beyond the Imitation Game: Quantifying and Extrapolating the Capabilities of Language Models. https://cogcomp.seas.upenn.edu/papers/SDGSRKUZCIQDSS23.pdf

[16] M. W. Bright, A. R. Hurson, and S. Pakzad. 1994. Automated resolution of semantic heterogeneity in multidatabases. *ACM Trans. Database Syst.* 19, 2 (June 1994), 212–253. doi:10.1145/176567.176569

[17] Alexander Brinkmann, Nick Baumann, and Christian Bizer. 2024. Using LLMs for the Extraction and Normalization of Product Attribute Values. In *Advances in Databases and Information Systems*, Joe Tekli, Johann Gamper, Richard Chbeir, and Yannis Manolopoulos (Eds.). Springer Nature Switzerland, Cham, 217–230.

[18] John Brooke. 1996. SUS-A quick and dirty usability scale. In *Usability Evaluation in Industry*, Patrick W. Jordan, Bruce Thomas, Ian Lyall McClelland, and Bernard Weerdmeester (Eds.). Taylor & Francis, London, 189–194.

[19] Zana Buçinca, Maja Barbara Malaya, and Krzysztof Z. Gajos. 2021. To Trust or to Think: Cognitive Forcing Functions Can Reduce Overreliance on AI in AI-assisted Decision-making. *Proceedings of the ACM on Human-Computer Interaction* 5, CSCW1, Article 188 (April 2021), 21 pages. doi:10.1145/3449287

[20] José Cambronero, Sumit Gulwani, Vu Le, Daniel Perelman, Arjun Radhakrishna, Clint Simon, and Ashish Tiwari. 2023. FlashFill++: Scaling Programming by Example by Cutting to the Chase. *Proceedings of the ACM on Programming Languages* 7, POPL, Article 33 (Jan. 2023), 30 pages. doi:10.1145/3571226

[21] Eswar Chandt. 2020. Amazon Movie Ratings. https://www.kaggle.com/datasets/eswarchandt/amazon-movie-ratings. Accessed: 2026-01-24.

[22] Shuaichen Chang and Eric Fosler-Lussier. 2023. How to Prompt LLMs for Text-to-SQL: A Study in Zero-shot, Single-domain, and Cross-domain Settings. arXiv:2305.11853 [cs.CL] https://arxiv.org/abs/2305.11853

[23] Carrie Chen. 2017. E-Commerce Data. https://www.kaggle.com/datasets/carrie1/ecommerce-data. Accessed: 2026-01-24.

[24] Ran Chen, Di Weng, Yanwei Huang, Xinhuan Shu, Jiayi Zhou, Guodao Sun, and Yingcai Wu. 2023. Rigel: Transforming Tabular Data by Declarative Mapping. *IEEE Transactions on Visualization and Computer Graphics* 29, 1 (jan 2023), 128–138. doi:10.1109/TVCG.2022.3209385 Conference Name: IEEE Transactions on Visualization and Computer Graphics.

[25] Wei-Hao Chen, Weixi Tong, Ph.D. Case, Amanda, and Tianyi Zhang. 2025. Dango: A Mixed-Initiative Data Wrangling System using Large Language Model. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems (CHI '25)*. Association for Computing Machinery, New York, NY, USA, Article 389, 28 pages. doi:10.1145/3706598.3714135

[26] Lidia Contreras-Ochando, Cèsar Ferri, José Hernández-Orallo, Fernando Martínez-Plumed, María José Ramírez-Quintana, and Susumu Katayama. 2019. Automated Data Transformation with Inductive Programming and Dynamic Background Knowledge. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2019, Würzburg, Germany, September 16–20, 2019, Proceedings, Part III* (Würzburg, Germany). Springer-Verlag, Berlin, Heidelberg, 735–751. doi:10.1007/978-3-030-46133-1_44

[27] Lidia Contreras-Ochando, César Ferri, José Hernández-Orallo, Fernando Martínez-Plumed, María José Ramírez-Quintana, and Susumu Katayama. 2018. General-purpose Declarative Inductive Programming with Domain-Specific Background Knowledge for Data Wrangling Automation. arXiv:1809.10054 [cs.AI] https://arxiv.org/abs/1809.10054

[28] Lidia Contreras-Ochando, Cèsar Ferri, José Hernández-Orallo, Fernando Martínez-Plumed, María José Ramírez-Quintana, and Susumu Katayama. 2020. Automated Data Transformation with Inductive Programming and Dynamic Background Knowledge. In *Machine Learning and Knowledge Discovery in Databases*, Ulf Brefeld, Elisa Fromont, Andreas Hotho, Arno Knobbe, Marloes Maathuis, and Céline Robardet (Eds.). Springer International Publishing, Cham, 735–751. doi:10.1007/978-3-030-46133-1_44

[29] Sanjib Das, AnHai Doan, Paul Suganthan G. C., Chaitanya Gokhale, Pradap Konda, Yash Govind, and Derek Paulsen. 2015. The Magellan Data Repository. https://sites.google.com/site/anhaidgroup/useful-stuff/the-magellan-data-repository. Accessed: 2026-01-24.

[30] Hong-Hai Do and Erhard Rahm. 2002. COMA: a system for flexible combination of schema matching approaches. In *Proceedings of the 28th International Conference on Very Large Data Bases*. VLDB Endowment, Hong Kong, China, 610–621.

[31] Peitong Duan, Jeremy Warner, and Bjoern Hartmann. 2023. Towards Generating UI Design Feedback with LLMs. In *Adjunct Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology* (San Francisco, CA, USA) *(UIST '23 Adjunct)*. Association for Computing Machinery, New York, NY, USA, Article 70, 3 pages. doi:10.1145/3586182.3615810

[32] David W. Embley, Li Xu, and Yihong Ding. 2004. Automatic direct and indirect schema mapping: experiences and lessons learned. *SIGMOD Rec.* 33, 4 (Dec. 2004), 14–19. doi:10.1145/1041410.1041413

[33] Sean M. Falconer and Natalya F. Noy. 2011. Interactive Techniques to Support Ontology Matching. In *Schema Matching and Mapping*, Zohra Bellahsene, Angela Bonifati, and Erhard Rahm (Eds.). Springer, Berlin, Heidelberg, 29–51. doi:10.1007/978-3-642-16518-4_2

[34] Xi Fang, Weijie Xu, Fiona Anting Tan, Jiani Zhang, Ziqing Hu, Yanjun Qi, Scott Nickleach, Diego Socolinsky, Srinivasan Sengamedu, and Christos Faloutsos. 2024. Large Language Models(LLMs) on Tabular Data: Prediction, Generation, and Understanding – A Survey. arXiv:2402.17944 [cs.CL] https://arxiv.org/abs/2402.17944

[35] Harald Foidl, Valentina Golendukhina, Rudolf Ramler, and Michael Felderer. 2024. Data pipeline quality: Influencing factors, root causes of data-related issues, and processing problem areas for developers. *Journal of Systems and Software* 207 (2024), 111855.

[36] Ian Funnell. 2025. Public vs Private LLMs: Secure AI for Enterprises. https://www.matillion.com/blog/public-vs-private-llms-enterprise-ai-security. Accessed: 2026-01-24.

[37] Avigdor Gal and Roee Shraga. 2022. Human's Role in-the-Loop. arXiv:2204.14192 [cs.DB] https://arxiv.org/abs/2204.14192

[38] Jie Gao, Simret Araya Gebreegziabher, Kenny Tsu Wei Choo, Toby Jia-Jun Li, Simon Tangi Perrault, and Thomas W Malone. 2024. A Taxonomy for Human-LLM Interaction Modes: An Initial Exploration. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI EA '24)*. Association for Computing Machinery, New York, NY, USA, Article 24, 11 pages. doi:10.1145/3613905.3650786

[39] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2024. Retrieval-Augmented Generation for Large Language Models: A Survey. arXiv:2312.10997 [cs.CL] https://arxiv.org/abs/2312.10997

[40] Roy Goldman and Jennifer Widom. 1997. DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases. In *Proceedings of the 23rd International Conference on Very Large Data Bases (VLDB '97)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 436–445.

[41] Stefan Grafberger, Paul Groth, Julia Stoyanovich, and Sebastian Schelter. 2022. Data distribution debugging in machine learning pipelines. *The VLDB Journal* 31, 5 (Jan. 2022), 1103–1126. doi:10.1007/s00778-021-00726-w

[42] Stefan Grafberger, Shubha Guha, Julia Stoyanovich, and Sebastian Schelter. 2021. MLINSPECT: A Data Distribution Debugger for Machine Learning Pipelines. In *Proceedings of the 2021 International Conference on Management of Data* (Virtual Event, China) *(SIGMOD '21)*. Association for Computing Machinery, New York, NY, USA, 2736–2739. doi:10.1145/3448016.3452759

[43] Ken Gu, Madeleine Grunde-McLaughlin, Andrew McNutt, Jeffrey Heer, and Tim Althoff. 2024. How Do Data Analysts Respond to AI Assistance? A Wizard-of-Oz Study. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '24)*. Association for Computing Machinery, New York, NY, USA, Article 1015, 22 pages. doi:10.1145/3613904.3641891

[44] Sumit Gulwani. 2011. Automating string processing in spreadsheets using input-output examples. In *Proceedings of the 38th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* (Austin, Texas, USA) *(POPL '11)*. Association for Computing Machinery, New York, NY, USA, 317–330. doi:10.1145/1926385.1926423

[45] Sumit Gulwani, William R. Harris, and Rishabh Singh. 2012. Spreadsheet data manipulation using examples. *Commun. ACM* 55, 8 (Aug. 2012), 97–105. doi:10.1145/2240236.2240260

[46] Sumit Gulwani and Mark Marron. 2014. NLyze: interactive programming by natural language for spreadsheet data analysis and manipulation. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data* (Snowbird, Utah, USA) *(SIGMOD '14)*. Association for Computing Machinery, New York, NY, USA, 803–814. doi:10.1145/2588555.2612177

[47] Philip J. Guo, Sean Kandel, Joseph M. Hellerstein, and Jeffrey Heer. 2011. Proactive wrangling: mixed-initiative end-user programming of data transformation scripts. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology* (Santa Barbara, California, USA) *(UIST '11)*. Association for Computing Machinery, New York, NY, USA, 65–74. doi:10.1145/2047196.2047205

[48] Laura M. Haas, Mauricio A. Hernández, Howard Ho, Lucian Popa, and Mary Roth. 2005. Clio grows up: from research prototype to industrial tool. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data* (Baltimore, Maryland) *(SIGMOD '05)*. Association for Computing Machinery, New York, NY, USA, 805–810. doi:10.1145/1066157.1066252

[49] William R. Harris and Sumit Gulwani. 2011. Spreadsheet table transformations from examples. In *Proceedings of the 32nd ACM SIGPLAN Conference on Programming Language Design and Implementation* (San Jose, California, USA) *(PLDI '11)*. Association for Computing Machinery, New York, NY, USA, 317–328. doi:10.1145/1993498.1993536

[50] Sandra G. Hart and Lowell E. Staveland. 1988. Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research. In *Human Mental Workload*, Peter A. Hancock and Najmedin Meshkati (Eds.). Advances in Psychology, Vol. 52. North-Holland, Amsterdam, 139–183. doi:10.1016/S0166-4115(08)62386-9

[51] Benjamin Hättasch, Michael Truong-Ngoc, Andreas Schmidt, and Carsten Binnig. 2022. It's AI Match: A Two-Step Approach for Schema Matching Using Embeddings. arXiv:2203.04366 [cs] doi:10.48550/arXiv.2203.04366

[52] Jeffrey Heer and Maneesh Agrawala. 2008. Design considerations for collaborative visual analytics. *Information Visualization* 7, 1 (March 2008), 49–62. doi:10.1145/1391107.1391112

[53] Stefan Hegselmann, Alejandro Buendia, Hunter Lang, Monica Agrawal, Xiaoyi Jiang, and David Sontag. 2023. TabLLM: Few-shot Classification of Tabular Data with Large Language Models. In *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 206)*. PMLR, Valencia, Spain, 5549–5581. https://proceedings.mlr.press/v206/hegselmann23a.html

[54] Zijin Hong, Zheng Yuan, Qinggang Zhang, Hao Chen, Junnan Dong, Feiran Huang, and Xiao Huang. 2025. Next-Generation Database Interfaces: A Survey of LLM-Based Text-to-SQL. *IEEE Transactions on Knowledge and Data Engineering* 37, 12 (2025), 7328–7345. doi:10.1109/TKDE.2025.3609486

[55] Sayed Hoseini, Andreas Burgdorf, Alexander Paulus, Tobias Meisen, Christoph Quix, and André Pomp. 2024. Towards LLM-augmented Creation of Semantic Models for Dataspaces. In *Proceedings of the Second International Workshop on Semantics in Dataspaces (SDS 2024) co-located with the 21st Extended Semantic Web Conference (ESWC 2024) (CEUR Workshop Proceedings, Vol. 3705)*. CEUR-WS.org, Aachen, Germany, 15 pages. https://ceur-ws.org/Vol-3705/

[56] Yanwei Huang, Yunfan Zhou, Ran Chen, Changhao Pan, Xinhuan Shu, Di Weng, and Yingcai Wu. 2024. Interactive Table Synthesis With Natural Language. *IEEE Transactions on Visualization and Computer Graphics* 30, 9 (Sept. 2024), 6130–6145. doi:10.1109/TVCG.2023.3329120

[57] Yanwei Huang, Yunfan Zhou, Ran Chen, Changhao Pan, Xinhuan Shu, Di Weng, and Yingcai Wu. 2024. Interactive Table Synthesis With Natural Language. *IEEE Transactions on Visualization and Computer Graphics* 30, 9 (sep 2024), 6130–6145. doi:10.1109/TVCG.2023.3329120 Conference Name: IEEE Transactions on Visualization and Computer Graphics.

[58] Petra Isenberg, Danyel Fisher, Sharoda A Paul, Meredith Ringel Morris, Kori Inkpen, and Mary Czerwinski. 2011. Co-located collaborative visual analytics around a tabletop display. *IEEE Transactions on visualization and Computer Graphics* 18, 5 (2011), 689–702. doi:10.1109/TVCG.2011.287

[59] H. V. Jagadish, Johannes Gehrke, Alexandros Labrinidis, Yannis Papakonstantinou, Jignesh M. Patel, Raghu Ramakrishnan, and Cyrus Shahabi. 2014. Big data and its technical challenges. *Commun. ACM* 57, 7 (July 2014), 86–94. doi:10.1145/2611567

[60] M. S. Jahid. 2024. Road Accident Trends in Bangladesh: 1980–2024. https://www.kaggle.com/datasets/msjahid/road-accident-statistics-in-bangladesh. Accessed: 2026-01-24.

[61] Gonzalo Jaimovitch-López, Cèsar Ferri, José Hernández-Orallo, Fernando Martínez-Plumed, and María José Ramírez-Quintana. 2023. Can language models automate data wrangling? *Machine Learning* 112, 6 (jun 2023), 2053–2082. doi:10.1007/s10994-022-06259-9

[62] Abhik Jha. 2019. MovieLens 100k. https://www.kaggle.com/datasets/abhikjha/movielens-100k. Accessed: 2026-01-24.

[63] Zhongjun Jin, Michael R. Anderson, Michael Cafarella, and H. V. Jagadish. 2017. Foofah: Transforming Data By Example. In *Proceedings of the 2017 ACM International Conference on Management of Data* (Chicago, Illinois, USA) *(SIGMOD '17)*. Association for Computing Machinery, New York, NY, USA, 683–698. doi:10.1145/3035918.3064034

[64] Kaggle. 2025. Find Open Datasets and Machine Learning Projects | Kaggle. https://www.kaggle.com/datasets. Accessed: 2026-01-24.

[65] Sean Kandel, Andreas Paepcke, Joseph Hellerstein, and Jeffrey Heer. 2011. Wrangler: interactive visual specification of data transformation scripts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vancouver, BC, Canada) *(CHI '11)*. Association for Computing Machinery, New York, NY, USA, 3363–3372. doi:10.1145/1978942.1979444

[66] Stephen Kasica, Charles Berret, and Tamara Munzner. 2020. Table Scraps: An Actionable Framework for Multi-Table Data Wrangling From An Artifact Study of Computational Journalism. http://arxiv.org/abs/2009.02373 arXiv:2009.02373 [cs].

[67] Anjali Khurana, Hariharan Subramonyam, and Parmit K Chilana. 2024. Why and When LLM-Based Assistants Can Go Wrong: Investigating the Effectiveness of Prompt-Based Interactions for Software Help-Seeking. In *Proceedings of the 29th International Conference on Intelligent User Interfaces* (Greenville, SC, USA) *(IUI '24)*. Association for Computing Machinery, New York, NY, USA, 288–303. doi:10.1145/3640543.3645200

[68] Henry F. Korth, Gabriel M. Kuper, Joan Feigenbaum, Allen van Gelder, and Jeffrey D. Ullman. 1984. SYSTEM/U: a database system based on the universal relation assumption. *ACM Trans. Database Syst.* 9, 3 (Sept. 1984), 331–347. doi:10.1145/1270.1209

[69] Christos Koutras, Marios Fragkoulis, Asterios Katsifodimos, and Christoph Lofi. 2020. REMA: Graph Embeddings-based Relational Schema Matching. In *Proceedings of the Workshops of the EDBT/ICDT 2020 Joint Conference (SEA Data Workshop) (CEUR Workshop Proceedings, Vol. 2578)*. CEUR-WS.org, Aachen, Germany, 4 pages. https://ceur-ws.org/Vol-2578/

[70] Christos Koutras, George Siachamis, Andra Ionescu, Kyriakos Psarakis, Jerry Brons, Marios Fragkoulis, Christoph Lofi, Angela Bonifati, and Asterios Katsifodimos. 2021. Valentine: Evaluating Matching Techniques for Dataset Discovery. In *Proceedings of the 37th IEEE International Conference on Data Engineering (ICDE)*. IEEE, Chania, Greece, 468–479. doi:10.1109/ICDE51399.2021.00047

[71] J.A. Larson, S.B. Navathe, and R. Elmasri. 1989. A Theory of Attributed Equivalence in Databases with Application to Schema Integration. *IEEE Transactions on Software Engineering* 15, 4 (April 1989), 449–463. doi:10.1109/32.16605

[72] Frank Legler and Felix Naumann. 2007. A Classification of Schema Mappings and Analysis of Mapping Tools. In *Datenbanksysteme in Business, Technologie und Web (BTW 2007) – 12. Fachtagung des GI-Fachbereichs "Datenbanken und Informationssysteme" (DBIS)*. Gesellschaft für Informatik e. V., Bonn, 449–463.

[73] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems* (Vancouver, BC, Canada) *(NIPS '20)*. Curran Associates Inc., Red Hook, NY, USA, Article 793, 16 pages.

[74] Guoliang Li, Xuanhe Zhou, and Xinyang Zhao. 2024. LLM for Data Management. *Proceedings of the VLDB Endowment* 17, 12 (Aug. 2024), 4213–4216. doi:10.14778/3685800.3685838

[75] Han Li, Yash Govind, Sidharth Mudgal, Theodoros Rekatsinas, and AnHai Doan. 2021. Deep Learning for Semantic Matching: A Survey. *Journal of Computer Science and Cybernetics* 37, 4 (Oct. 2021), 365–402. doi:10.15625/1813-9663/37/4/16151

[76] Steve Lohr. 2014. For big-data scientists,'janitor work'is key hurdle to insights. *New York Times* 17 (2014), B4.

[77] Jayant Madhavan, Philip A. Bernstein, and Erhard Rahm. 2001. Generic Schema Matching with Cupid. In *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB '01)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 49–58.

[78] Potsawee Manakul, Adian Liusie, and Mark Gales. 2023. SelfCheckGPT: Zero-Resource Black-Box Hallucination Detection for Generative Large Language

Models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, Singapore, 9004–9017. doi:10.18653/v1/2023.emnlp-main.557

[79] Yaoli Mao, Dakuo Wang, Michael Muller, Kush R. Varshney, Ioana Baldini, Casey Dugan, and Aleksandra Mojsilović. 2019. How Data ScientistsWork Together With Domain Experts in Scientific Collaborations: To Find The Right Answer Or To Ask The Right Question? *Proceedings of the ACM on Human-Computer Interaction* 3, GROUP, Article 237 (Dec. 2019), 23 pages. doi:10.1145/3361118

[80] Marian447. 2021. Retail Store Sales Transactions (Scanner Data). https://www.kaggle.com/datasets/marian447/retail-store-sales-transactions. Accessed: 2026-01-24.

[81] Bruno Marnette, Giansalvatore Mecca, Paolo Papotti, Salvatore Raunich, and Donatello Santoro. 2011. ++Spicy: an open-source tool for second-generation schema mapping and data exchange. *Proceedings of the VLDB Endowment* 4, 12 (Aug. 2011), 1438–1441. doi:10.14778/3402755.3402790

[82] Mikaël Mayer, Gustavo Soares, Maxim Grechkin, Vu Le, Mark Marron, Oleksandr Polozov, Rishabh Singh, Benjamin Zorn, and Sumit Gulwani. 2015. User Interaction Models for Disambiguation in Programming by Example. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology* (Charlotte, NC, USA) *(UIST '15)*. Association for Computing Machinery, New York, NY, USA, 291–301. doi:10.1145/2807442.2807459

[83] Sergey Melnik, Hector Garcia-Molina, and Erhard Rahm. 2002. Similarity Flooding: A Versatile Graph Matching Algorithm and Its Application to Schema Matching. In *Proceedings of the 18th International Conference on Data Engineering (ICDE '02)*. IEEE Computer Society, USA, 117.

[84] Microsoft. 2021. Using BizTalk Mapper — BizTalk Server. https://learn.microsoft.com/en-us/biztalk/core/using-biztalk-mapper. Accessed: 2026-01-24.

[85] Microsoft. 2025. Microsoft Copilot for Microsoft 365 | Microsoft 365. https://www.microsoft.com/en-us/microsoft-365/enterprise/copilot-for-microsoft-365. Accessed: 2026-01-24.

[86] Renée J. Miller, Laura M. Haas, and Mauricio A. Hernández. 2000. Schema Mapping as Query Discovery. In *Proceedings of the 26th International Conference on Very Large Data Bases (VLDB '00)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 77–88.

[87] Michael Muller, Ingrid Lange, Dakuo Wang, David Piorkowski, Jason Tsay, Q. Vera Liao, Casey Dugan, and Thomas Erickson. 2019. How Data Science Workers Work with Data: Discovery, Capture, Curation, Design, Creation. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–15. doi:10.1145/3290605.3300356

[88] Clifford Nass and Scott Brave. 2007. *Wired for Speech: How Voice Activates and Advances the Human-Computer Relationship*. The MIT Press, Cambridge, MA, USA.

[89] Alfredo Nazabal, Christopher K. I. Williams, Giovanni Colavizza, Camila Rangel Smith, and Angus Williams. 2020. Data Engineering for Data Analytics: A Classification of the Issues, and Case Studies. doi:10.48550/arXiv.2004.12929 arXiv:2004.12929 [cs].

[90] Olist. 2021. Brazilian E-Commerce Public Dataset by Olist. https://www.kaggle.com/datasets/olistbr/brazilian-ecommerce. Accessed: 2026-01-24.

[91] Samir Passi and Steven J. Jackson. 2018. Trust in Data Science: Collaboration, Translation, and Accountability in Corporate Data Science Projects. *Proceedings of the ACM on Human-Computer Interaction* 2, CSCW, Article 136 (Nov. 2018), 28 pages. doi:10.1145/3274405

[92] Norman W. Paton. 2019. Automating Data Preparation: Can We? Should We? Must We?. In *Proceedings of the 21st International Workshop on Design, Optimization, Languages and Analytical Processing of Big Data (DOLAP 2019), co-located with EDBT/ICDT 2019 (CEUR Workshop Proceedings, Vol. 2324)*. CEUR-WS.org, Aachen, Germany, 5] pages. https://ceur-ws.org/Vol-2324/

[93] Alejo Paullier. 2022. New York City Weather Data 2019. https://www.kaggle.com/datasets/alejopaullier/new-york-city-weather-data-2019. Accessed: 2026-01-24.

[94] Kathleen H. Pine and Max Liboiron. 2015. The Politics of Measurement and Action. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (Seoul, Republic of Korea) *(CHI '15)*. Association for Computing Machinery, New York, NY, USA, 3147–3156. doi:10.1145/2702123.2702298

[95] Meikel Poess, Tilmann Rabl, Hans-Arno Jacobsen, and Brian Caufield. 2014. TPC-DI: the first industry benchmark for data integration. *Proceedings of the VLDB Endowment* 7, 13 (Aug. 2014), 1367–1378. doi:10.14778/2733004.2733009

[96] Qlik. 2025. Talend Data Integration — Software to Connect, Access, and Transform Data. https://www.talend.com/products/integrate-data/. Talend was acquired by Qlik. Accessed: 2026-01-24.

[97] Rachit239. 2023. Road Accident Data 2020 India. https://www.kaggle.com/datasets/rachit239/road-accident-data-2020-india. Accessed: 2026-01-24.

[98] Alessandro Raffio, Daniele Braga, Stefano Ceri, Paolo Papotti, and Mauricio A. Hernandez. 2008. Clip: a Visual Language for Explicit Schema Mappings. In *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering (ICDE '08)*. IEEE Computer Society, USA, 30–39. doi:10.1109/ICDE.2008.4497411

[99] Erhard Rahm and Philip A. Bernstein. 2001. A survey of approaches to automatic schema matching. *The VLDB Journal* 10, 4 (Dec. 2001), 334–350. doi:10.1007/s007780100057

[100] Vijayshankar Raman and Joseph M. Hellerstein. 2000. *An Interactive Framework for Data Cleaning*. Technical Report. University of California at Berkeley, USA.

[101] Vijayshankar Raman and Joseph M. Hellerstein. 2001. Potter's Wheel: An Interactive Data Cleaning System. In *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB '01)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 381–390.

[102] Mark Ryan. 2020. In AI We Trust: Ethics, Artificial Intelligence, and Reliability. *Science and Engineering Ethics* 26, 5 (2020), 2749–2767. doi:10.1007/s11948-020-00228-y

[103] Nabeel Seedat and Mihaela van der Schaar. 2024. Matchmaker: Self-Improving Large Language Model Programs for Schema Matching. arXiv:2410.24105 [cs] doi:10.48550/arXiv.2410.24105

[104] Harshit Shankhdhar. 2021. IMDB Dataset of Top 1000 Movies and TV Shows. https://www.kaggle.com/datasets/harshitshankhdhar/imdb-dataset-of-top-1000-movies-and-tv-shows. Accessed: 2026-01-24.

[105] Eitam Sheetrit, Menachem Brief, Moshik Mishaeli, and Oren Elisha. 2024. ReMatch: Retrieval Enhanced Schema Matching with LLMs. arXiv:2403.01567 [cs] doi:10.48550/arXiv.2403.01567

[106] Ben Shneiderman. 2007. Creativity support tools: accelerating discovery and innovation. *Commun. ACM* 50, 12 (Dec. 2007), 20–32. doi:10.1145/1323688.1323689

[107] Rishabh Singh. 2016. BlinkFill: semi-supervised programming by example for syntactic string transformations. *Proceedings of the VLDB Endowment* 9, 10 (June 2016), 816–827. doi:10.14778/2977797.2977807

[108] Skyflow. 2024. Private LLMs: Data Protection Potential and Limitations. https://www.skyflow.com/post/private-llms-data-protection-potential-and-limitations. Accessed: 2026-01-24.

[109] Stylus Studio. 2025. XML Editor, XML Tools, and XQuery — Stylus Studio. https://www.stylusstudio.com/. Accessed: 2026-01-24.

[110] Hari Subramonyam, Roy Pea, Christopher Pondoc, Maneesh Agrawala, and Colleen Seifert. 2024. Bridging the Gulf of Envisioning: Cognitive Challenges in Prompt Based Interactions with LLMs. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '24)*. Association for Computing Machinery, New York, NY, USA, Article 1039, 19 pages. doi:10.1145/3613904.3642754

[111] The Devastator. 2022. E-Commerce Sales Dataset. https://www.kaggle.com/datasets/thedevastator/unlock-profits-with-e-commerce-sales-data. Accessed: 2026-01-24.

[112] TMDB. 2017. TMDB 5000 Movie Dataset. https://www.kaggle.com/datasets/tmdb/tmdb-movie-metadata. Accessed: 2026-01-24.

[113] Helena Vasconcelos, Matthew Jörke, Madeleine Grunde-McLaughlin, Tobias Gerstenberg, Michael S. Bernstein, and Ranjay Krishna. 2023. Explanations Can Reduce Overreliance on AI Systems During Decision-Making. *Proceedings of the ACM on Human-Computer Interaction* 7, CSCW1, Article 129 (April 2023), 38 pages. doi:10.1145/3579605

[114] Panos Vassiliadis. 2009. A Survey of Extract–Transform–Load Technology. *International Journal of Data Warehousing and Mining (IJDWM)* 5, 3 (July 2009), 1–27. doi:10.4018/jdwm.2009070101

[115] Amy Voida, Ellie Harmon, and Ban Al-Ani. 2011. Homebrew databases: complexities of everyday information management in nonprofit organizations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vancouver, BC, Canada) *(CHI '11)*. Association for Computing Machinery, New York, NY, USA, 915–924. doi:10.1145/1978942.1979078

[116] Chenglong Wang, Bongshin Lee, Steven M. Drucker, Dan Marshall, and Jianfeng Gao. 2025. Data Formulator 2: Iterative Creation of Data Visualizations, with AI Transforming Data Along the Way. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems (CHI '25)*. Association for Computing Machinery, New York, NY, USA, Article 677, 17 pages. doi:10.1145/3706598.3713296

[117] Dakuo Wang, Josh Andres, Justin D. Weisz, Erick Oduor, and Casey Dugan. 2021. AutoDS: Towards Human-Centered Automation of Data Science. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 79, 12 pages. doi:10.1145/3411764.3445526

[118] Qiu Yue Wang, Jeffrey X. Yu, and Kam-Fai Wong. 2000. Approximate Graph Schema Extraction for Semi-Structured Data. In *Proceedings of the 7th International Conference on Extending Database Technology: Advances in Database Technology (EDBT '00)*. Springer-Verlag, Berlin, Heidelberg, 302–316.

[119] Syed Mekael Wasti, Ken Q. Pu, and Ali Neshati. 2024. Large Language User Interfaces: Voice Interactive User Interfaces Powered by LLMs. In *Intelligent Systems and Applications*, Kohei Arai (Ed.). Springer Nature Switzerland, Cham, 639–655.

[120] Emmanuel F. Werr. 2022. London Weather Data. https://www.kaggle.com/datasets/emmanuelfwerr/london-weather-data. Accessed: 2026-01-24.

[121] Jacob O. Wobbrock, Leah Findlater, Darren Gergle, and James J. Higgins. 2011. The aligned rank transform for nonparametric factorial analyses using only anova procedures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vancouver, BC, Canada) *(CHI '11)*. Association for Computing Machinery, New York, NY, USA, 143–146. doi:10.1145/1978942.1978963

[122] Liwenhan Xie, Chengbo Zheng, Haijun Xia, Huamin Qu, and Chen Zhu-Tian. 2024. WaitGPT: Monitoring and Steering Conversational LLM Agent in Data Analysis with On-the-Fly Code Visualization. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology* (Pittsburgh, PA, USA) *(UIST '24)*. Association for Computing Machinery, New York, NY, USA, Article 119, 14 pages. doi:10.1145/3654777.3676374

[123] J.D. Zamfirescu-Pereira, Shm Garanganao Almeda, Kyu Won Kim, and Bjoern Hartmann. 2023. Towards Image Design Space Exploration in Spreadsheets with LLM Formulae. In *Adjunct Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology* (San Francisco, CA, USA) *(UIST '23 Adjunct)*. Association for Computing Machinery, New York, NY, USA, Article 82, 3 pages. doi:10.1145/3586182.3615790

[124] J.D. Zamfirescu-Pereira, Richmond Y. Wong, Bjoern Hartmann, and Qian Yang. 2023. Why Johnny Can't Prompt: How Non-AI Experts Try (and Fail) to Design LLM Prompts. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI '23)*. Association for Computing Machinery, New York, NY, USA, Article 437, 21 pages. doi:10.1145/3544548.3581388

[125] Amy X. Zhang, Michael Muller, and Dakuo Wang. 2020. How do Data Science Workers Collaborate? Roles, Workflows, and Tools. *Proceedings of the ACM on Human-Computer Interaction* 4, CSCW1, Article 22 (May 2020), 23 pages. doi:10.1145/3392826

[126] Michael JQ Zhang and Eunsol Choi. 2025. Clarify When Necessary: Resolving Ambiguity Through Interaction with LMs. In *Findings of the Association for Computational Linguistics: NAACL 2025*, Luis Chiruzzo, Alan Ritter, and Lu Wang (Eds.). Association for Computational Linguistics, Albuquerque, New Mexico, 5526–5543. doi:10.18653/v1/2025.findings-naacl.306

[127] Yunjia Zhang, Avrilia Floratou, Joyce Cahoon, Subru Krishnan, Andreas C. Müller, Dalitso Banda, Fotis Psallidas, and Jignesh M. Patel. 2023. Schema Matching using Pre-Trained Language Models. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, Anaheim, CA, USA, 1558–1571. doi:10.1109/ICDE55515.2023.00123

[128] Yunjia Zhang, Jordan Henkel, Avrilia Floratou, Joyce Cahoon, Shaleen Deep, and Jignesh M. Patel. 2024. ReAcTable: Enhancing ReAct for Table Question Answering. *Proceedings of the VLDB Endowment* 17, 8 (April 2024), 1981–1994. doi:10.14778/3659437.3659452

[129] Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, Longyue Wang, Anh Tuan Luu, Wei Bi, Freda Shi, and Shuming Shi. 2025. Siren's Song in the AI Ocean: A Survey on Hallucination in Large Language Models. *Computational Linguistics* 51, 4 (12 2025), 1373–1418. arXiv:https://direct.mit.edu/coli/article-pdf/51/4/1373/2535477/coli.a.16.pdf doi:10.1162/COLI.a.16

A Technical Evaluation Details

A.1 Benchmark Dataset Construction Methodology

Existing schema matching datasets (e.g., Magellan [29], TPC-DI [95], Wikidata [70]) focus on one-to-one element correspondences and lack the complex transformation operations needed for disjoint dataset conversion, which has been a longstanding challenge in schema matching systems [1, 10]. To evaluate our system's capacity for inferring non-trivial transformations from disjoint datasets, we curated two problem sets: (a) *isolated problems* testing individual transformation operations, and (b) *compound problems* using real dataset pairs from Kaggle [64] requiring multiple transformation steps. Our benchmark is made open source[1].

Table A1: Categorization of data transformation operations used in our isolated problem set, along with the count of problem variants for each transformation type, and the total number of transformations required to complete the data conversion.

Modification	Transformation Type	Variants	Trans. Count	Problem Description
Structural Modification	Renaming Columns	3	14	Align source column names with spec dataset. **e.g.,** "temp" to "Temperature"; "Salario Anual" to "Annual Salary"
	Deleting Extra Columns	2	5	Remove columns not present in spec dataset. **e.g.,** removing "Middle Name" column
	Reordering Columns	1	1	Match column order to spec dataset. **e.g.,** reordering columns Name, Roll to Roll, Name
	Sorting Data	2	3	Arrange rows to match spec dataset order. **e.g.,** sort by "date"; sort by "dept", then by "name"
	Merging Data	3	18	Combine multiple source files to match spec dataset structure. **e.g.,** vertical merge; horizontal merge; merge with tag
	Splitting Data	2	6	Divide source data into multiple files to align with spec dataset. **e.g.,** divide dataset by "organization" into multiple CSVs
	Data Consolidation	2	6	Join multiple columns to match spec dataset structure. **e.g.,** join with delimeter; consolidate in JSON structure
	Data Segregation	2	18	Split single columns into multiple to align with spec dataset. **e.g.,** split by delimeter; expand JSON structure into columns
	Pivot Data	2	2	Transform data between long and wide formats to match spec dataset. **e.g.,** unfolding columns for individual quarter sales to "quarter" and "sales"
	Transpose Data	1	1	Switch rows/columns to match spec dataset layout. **e.g.,** changing layout from respondent-wise responses to question-wise responses
Content Modification	Data Unit Conversion	4	17	Adjust data units to match spec dataset requirements. **e.g.,** converting Imperial units to Metric
	Data Format Conversion	2	6	Alter data representation to match spec dataset. **e.g.,** changing date format from MM/DD/YYYY to YYMMDD
	Data Filtering	3	4	Remove data from source based on some spec dataset criterion. **e.g.,** value-based/categorical/multi-conditional filtering
	Data Aggregation	3	11	Summarize multiple rows to match spec dataset level. **e.g.,** calculating sum, average, or standard deviation
	Data Binning	1	5	Group source data into categories matching spec dataset. **e.g.,** grouping ages into ranges (0-18, 19-35, etc.)
	Categorical Remapping	3	5	Reassign categories to align with spec dataset. **e.g.,** mapping Male/Female to "M"/"F"; one-hot encoding
	Data Standardization	4	18	Standardize unstructured data to match spec dataset requirements. **e.g.,** converting unstructured full addresses into street, country, zip
	Data Derivation	3	6	Perform logical operations to derive new data from source. **e.g.,** calculating age from DOB; sentiment from text comment

Table A2: Compound Problem Set: Real world dataset pairs used in the technical evaluation.

Problem Category	Source/Spec Columns	Non-Derivable Spec Columns	Transformation Count	Dataset References
Weather	10 / 10	4	9	**Source**:*New York City Weather Data 2019* [93] **Spec**:*London Weather Data* [120]
Inventory	10 / 8	1	7	**Source**: *E-Commerce Sales Dataset* [111] **Spec**:*Retail Store Sales Transactions* [80]
Movies-1	207 / 4	1	2	**Source**:*Amazon Movie Ratings* [21] **Spec**:*MovieLens 100k* [62]
Movies-2	20 / 16	8	9	**Source**:*TMDB 5000 Movie Dataset* [112] **Spec**:*IMDB Movies Dataset* [104]
Accidents	5 / 7	2	4	**Source**:*Road Accident Data 2020 India* [97] **Spec**:*Road Accident Trends in Bangladesh: 1980-2024* [60]

[1]https://adildsw.com/semantics-dataset/

Isolated Problem Set: We reviewed prior work on data transformation and wrangling [15, 17, 24, 45, 57, 61, 63, 65, 89, 100, 101] to compile a comprehensive set of operations relevant to data conversion scenarios. Following Gulwani et al.'s categorization of transformations into *Syntactic*, *Semantic*, and *Layout* types [45], we synthesized operation types from data wrangling tools including Potter's Wheel [101], Wrangler [65], Foofah [63], and recent frameworks [24, 57, 89]. We grouped transformations into two categories: (a) *Structural Modifications* (11 types): Operations altering data layout or organization, including merging, splitting, pivoting, column addition/removal, row filtering, and reshaping operations; (b) *Content Modifications* (8 types): Operations altering data values or semantics, including aggregation, derivation, unit conversion, format standardization, categorical remapping, and semantic extraction

We defined multiple variants for each transformation type to capture real-world diversity, yielding 43 total transformation tasks. For each task, we manually designed both source and specification schemas to reflect the intended transformation, ensuring diverse complexity levels. We used GPT-4 (via OpenAI chat interface) solely for generating synthetic data values for these pre-designed schemas, with explicit constraints preventing any entity or value overlap between source and target datasets. GPT-4's role was limited strictly to data value generation; it did not contribute to the design or formulation of the transformation tasks themselves. Table A1 summarizes this problem set, with complete specifications in Table A3.

Compound Problem Set: To evaluate real-world conversion scenarios requiring multiple chained operations, we collected 10 publicly available datasets from Kaggle [64], paired across four domains (movies, weather, e-commerce, accident reporting). These dataset pairs exhibit the structural, semantic, and granularity differences typical of real-world data integration challenges, including non-derivable attributes requiring human input. Problem pairs are detailed in Table A2.

A.2 Technical Evaluation Results

Figure A1 illustrates the evaluation results, showing transformation accuracy and confidence levels for both isolated and compound problem sets.

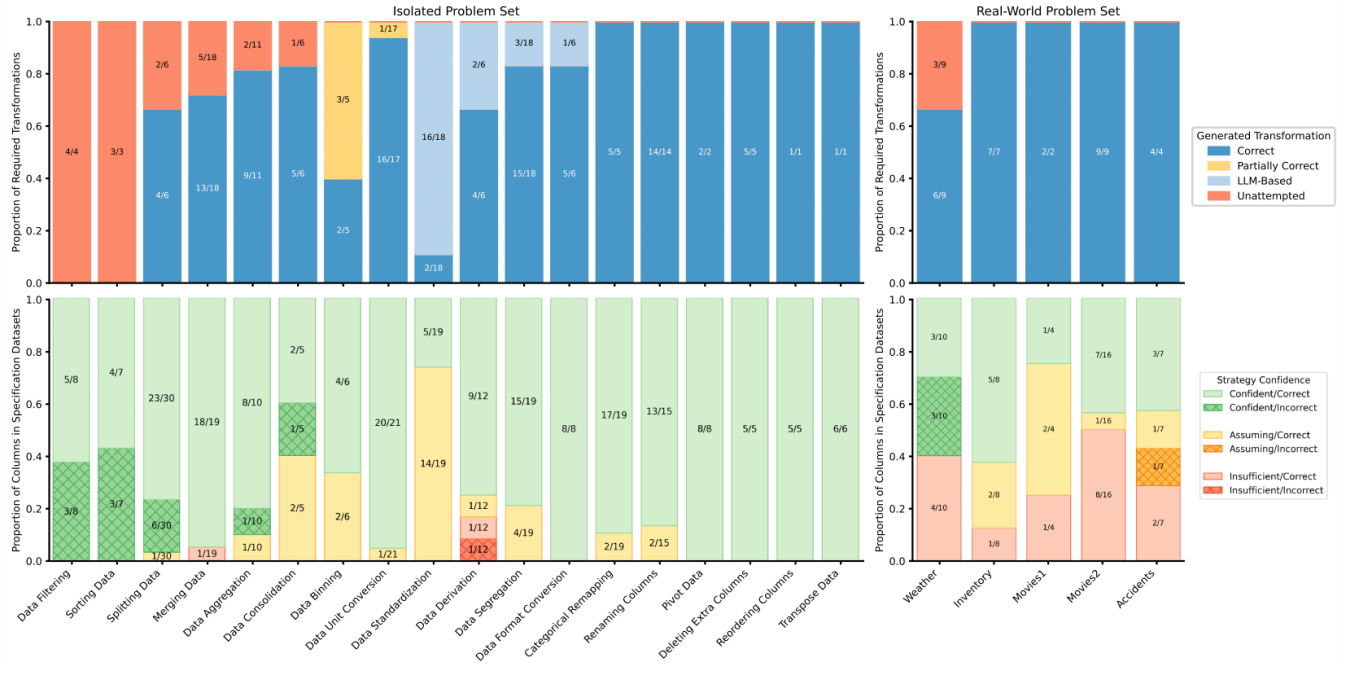

Figure A1: Technical evaluation results for isolated (left) and real-world (right) problem sets. Top: Transformation accuracy. Bottom: Confidence calibration analysis, jointly presenting success rates stratified by confidence level to assess whether interventions trigger appropriately. The system demonstrates well-calibrated confidence: confident classifications achieve 93% (isolated) and 86% (real-world) accuracy, while assuming and insufficient classifications reliably surface cases requiring human input—indicating low risk of the system 'failing confidently.'

A.3 Technical Evaluation Supplementary Materials

Table A3 presents a summarized breakdown of the transformation types and their variants, and Figure A2 shows example problem sets along with representative transformation code snippets generated by DataSpeck. The full set of problem definitions, ground-truth transformation steps for each task, and DataSpeck's generated outputs are provided separately in the supplementary materials accompanying this paper.

Figure A2: Examples of our Isolated Problem Sets used in the technical evaluation along with code snippets demonstrating the essential transformations generated by DataSpeck.

Table A3: Detailed breakdown of the isolated problem set used in our technical evaluation. Each row corresponds to a specific transformation task, including its category, variant, schema characteristics (number of source/spec files and columns), and the number of transformation steps required. This table expands on the summarized categorization presented in Table A1.

ID	Category	Problem	Variant	Src Files	Spec Files	Total Src Columns	Total Spec Columns	Trans. Count
1	Structural Modifications	Renaming Columns	Basic Renaming	1	1	2	2	1
2			Semantic Renaming	1	1	3	3	3
3			Language Renaming	1	1	10	10	10
4		Deleting Extra Columns	Pruning 1	1	1	6	3	3
5			Pruning 2	1	1	7	2	2
6		Reordering Columns	Reordering 1	1	1	5	5	1
7		Sorting Data	Single Column Sorting	1	1	3	3	1
8			Multi-Column Sorting	1	1	4	4	2
9		Merging Data	Vertical Merge	5	1	25	5	5
10			Vertical Merge With ID	5	1	25	6	10
11			Horizontal Merge	3	1	10	8	3
12		Splitting Data	Categorical Split 1	1	3	5	15	3
13			Categorical Split 2	1	3	5	15	3
14		Data Consolidation	Delimeter Join	1	1	9	3	3
15			Structured Consolidation	1	1	10	2	3
16		Data Segregation	Delimeter Split	1	1	3	9	9
17			Structured Segregation	1	1	2	10	9
18		Pivot Data	Wide to Long	1	1	5	3	1
19			Long to Wide	1	1	3	5	1
20		Transpose Data	Transpose 1	1	1	21	6	1
21	Content Modifications	Data Unit Conversion	Weather	1	1	5	5	4
22			Product	1	1	7	7	6
23			Transportation	1	1	5	5	4
24			Construction	1	1	4	4	3
25		Data Format Conversion	Format 1	1	1	4	4	3
26			Format 2	1	1	4	4	3
27		Data Filtering	Date-based Filtering	1	1	4	4	1
28			Category-based Filtering	1	1	3	2	1
29			Multi-column Filtering	1	1	3	2	2
30		Data Aggregation	Summing	1	1	3	2	2
31			Averaging	1	1	2	2	3
32			Statistical Aggregation	1	1	3	6	6
33		Data Binning	Binning Tasks	1	1	6	6	5
34		Categorical Remapping	Standard Remapping	1	1	4	4	3
35			One-Hot Encoding	1	1	5	10	1
36			Reverse One-Hot	1	1	10	5	1
37		Data Standardization	Measurement Standard.	1	1	4	4	4
38			Address Standardization	1	1	2	6	5
39			Date and Time Standard.	1	1	1	2	2
40			Personal Standardization	1	1	2	7	7
41		Data Derivation	Logical Derivation	1	1	6	4	2
42			Inter-Row Derivation	1	1	3	4	2
43			Semantic Derivation	1	1	3	4	2

B User Study Tasks

The following tables and task descriptions were used in our user study and are included here as supplementary material. They contain the exact datasets, summaries, and instructions provided to participants to guide each task.

B.1 Task 1/Variant A: School System Migration

Scenario: You are the record manager in a school, and your system recently updated to streamline the data storage pipeline. However, the new data structure is different from the previously collected data, and now the previous data cannot be loaded in the new system. Your task is to convert the old data into the format of the new data so that they can be loaded in the new data storage pipeline.

Grade Calculation Range: Grade < 50: F; 50 < Grade <= 100: D; 100 < Grade <= 200: C; 200 < Grade <= 300: B; 300 < Grade : A

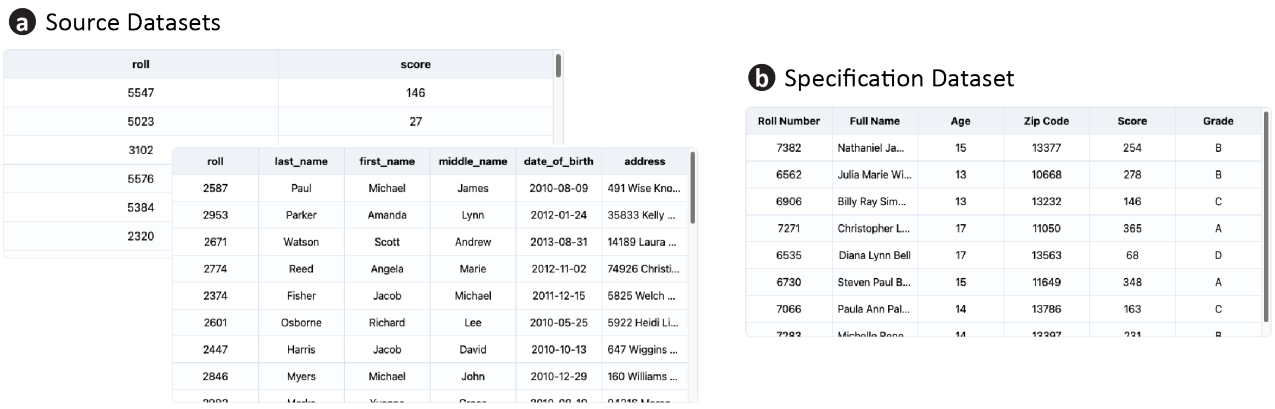

Figure B1: Task 1/Variant A: School System Migration

Additional Task: This new streamlined dataset does not contain students who have received Grade F

B.2 Task 1/Variant B: Warehouse Inventory Migration

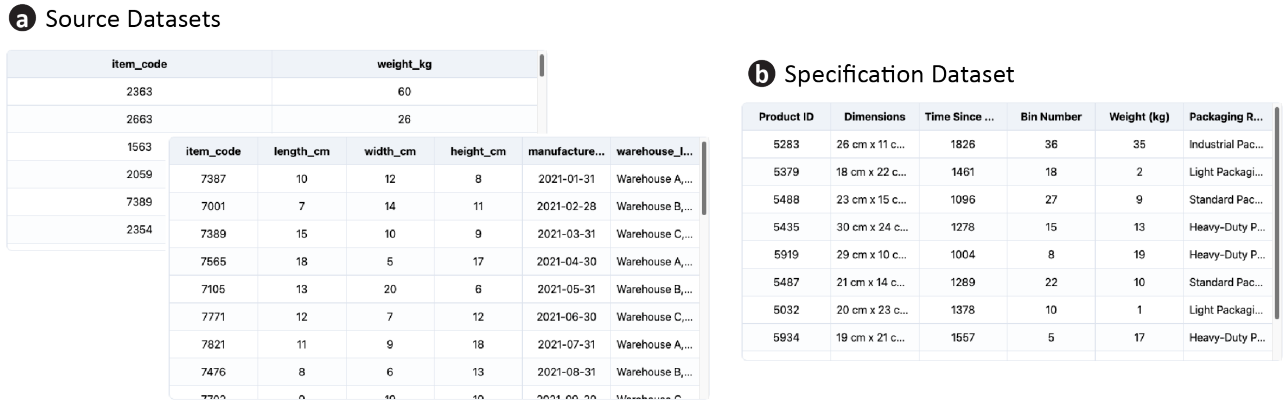

Figure B2: Task 1/Variant B: Warehouse Inventory Migration

Scenario: You are the record manager in a warehouse, and your system recently updated to streamline the data storage pipeline. However, the new data structure is different from the previously collected data, and now the previous data cannot be loaded in the new system. Your task is to convert the old data into the format of the new data so that they can be loaded in the new data storage pipeline.

Grade Calculation Range: Weight <= 5kg: Light Packaging; 5kg < Weight <= 10kg: Standard Packaging; 10kg < Weight <= 20kg: Heavy-Duty Packaging; 20kg < Weight <= 40kg: Industrial Packaging; 40kg < Requires Specialized Equipment

Additional Task: This new streamlined dataset only contains items that do not require specialized equipment.

B.3 Task 2/Variant A: Restructuring Plant Features

Scenario: You are a botanical researcher that works with summarized feature data for various plant genus/species. You typically receive data that is already summarized. However, you are now required to work with some raw feature data which contains readings for various other features for various genus/species. To successfully load this new raw data into your system, it must be converted into the summarized version that your system supports.

Note: This dataset comes from a different source and does not contain the same features present in the specification dataset

a Source Dataset

reading_nu...	Plant	Plant Height	Leaf Area	Leaf Nitrogen	Leaf Mass	Seed Mass
1	Fagus sylvatica	3.96	2.8	6.68	12.56	12.64
2	Pinus strobus	6.29	7.12	0.81	9.24	14.86
3	Carya ovata	3.23	12.68	-	5.77	-
4	Fagus sylvatica	13.92	-	10.19	12.69	7.09
5	Acer sacchar...	-	3.02	2.57	-	-
6	Quercus robur	0.72	-	-	10.03	10.46
7	Fraxinus exc...	13.55	1.15	0.77	5.75	-

b Specification Dataset

feature_id	feature	mean_reading	max_reading	max_readin...	reading_count
1	Root Length	18.52	24.94	Sansevieria tri...	53
2	Stem Thickness	11.73	23.57	Monstera deli...	64
3	Chlorophyll C...	16.43	29.06	Ficus lyrata	32
4	Water Uptake ...	11.64	42.75	Sansevieria tri...	72
5	Nutrient Abso...	14.7	34.78	Monstera deli...	23

Figure B3: Task 2/Variant A: Restructuring Plant Features

B.4 Task 2/Variant B: Restructuring Weather Data

a Source Dataset

weather_id	date	London	Kolkata	Sydney	Tokyo	Athens
1	1/1/2018	1.6	-	-	-	12.6
2	1/4/2018	1.9	-	18.8	-	-
3	1/6/2018	-2	-	-	9.2	7.1
4	1/8/2018	4.4	-	-	4.9	-
5	2/26/2018	4.5	-	24.6	-	11.9
6	2/27/2018	2	17.6	24	8.2	6.5
7	3/2/2018	6.3	-	-	-	-

b Specification Dataset

city_id	city	median_temp	min_temp	coldest_date	data_count
1	New York	17.3	-3.2	2/4/2018	20
2	Los Angeles	13.6	-12.9	2/12/2018	42
3	Chicago	19	-4.3	1/9/2018	12
4	Houston	11.7	-8.1	12/22/2018	94
5	Phoenix	11.1	-7.3	12/5/2018	32
6	Philadelphia	14.7	-6	2/17/2018	33
7	San Antonio	6.6	-17.2	12/9/2018	25
8	San Diego	10.7	-10.2	12/10/2018	54

Figure B4: Task 2/Variant B: Restructuring Weather Data

Scenario: You are a weather analyst that works with summarized weather data for various cities. You typically receive data that is already summarized. However, you are now required to work with some raw weather data which contains temperature readings for various cities for various dates. To successfully load this new raw data into your system, it must be converted into the summarized version that your system supports.

Note: This dataset comes from a different source and does not contain the cities present in the specification dataset

C DataSpeck System Prompts

The prompts for the various LLM-based processes of DataSpeck, such as the **Semantic Descriptor Analyser**, **Semantic Relationship Analyzer**, **Transform Step Generator**, and **Confidence Evaluator** are provided separately in the supplementary materials accompanying this paper.