CLUE: Answering How-To Questions on Dashboards using Metric Lineage Extractor and Simulator

Wei-Xuan Wang
University of Utah
Salt Lake City, Utah, USA
elisa.wang@utah.edu

Whanhee Cho
University of Utah
Salt Lake City, Utah, USA
whanhee.cho@utah.edu

Mahmood Jasim
Louisiana State University
Baton Rouge, Louisiana
USA
mjasim@lsu.edu

Anna Fariha
University of Utah
Salt Lake City, Utah, USA
afariha@cs.utah.edu

ABSTRACT

Data visualization on dashboards is ubiquitous across many domains, including business intelligence, healthcare analytics, finance, education, scientific research, and government and public policy. Such dashboards often display final metrics (e.g., revenue) that are derived from a series of data transformations and aggregations involving sub-metrics (e.g., sales, cost). However, users frequently struggle to understand these metrics, particularly when seeking answers to `how-to` questions, such as how to achieve a target metric value by adjusting the underlying data. This lack of transparency hinders users' ability to make informed decisions, since it is often unclear how metrics are computed and which factors contribute to changes in their values. We present CLUE, a system that provides an intuitive visualization of the lineage behind metric computation—from data extraction and transformation to aggregation and final computation—enabling users to understand the data and transformation processes transparently. Additionally, CLUE includes a *metric simulator* for addressing `how-to` questions: users can input their desired final and sub-metric values, and the system provides data adjustment suggestions to help achieve those targets. We demonstrate CLUE within a dashboard with multiple metrics over a real-world dataset.

1 INTRODUCTION

Data visualizations on dashboards are ubiquitous across industries and play a central role in supporting decision-making at all organizational levels. Such visualizations are often driven by complex data pipelines, where data from multiple sources is integrated and transformed using operations such as joins, filters, and aggregations. Through a series of transformations, raw data is converted into business metrics on dashboards, enabling organizations to summarize and track key performance indicators such as revenue, conversion rate, and customer lifetime value. These metrics are essential for monitoring business health, identifying trends, and making informed decisions. However, the underlying data and intermediate transformation are often hidden, as these visualizations typically present only the final aggregated metrics. This lack of transparency into how metrics are computed leads to challenges for users unfamiliar with them, causing them to misinterpret trends, struggle to diagnose anomalies, or lose trust in the reported numbers [7].

A common way users engage with dashboard metrics is through goal-driven `how-to` questions such as "*How can we increase this metric?*". However, without a clear understanding of how the metric is calculated in the first place, these questions become difficult to answer. Data lineage and provenance systems [4, 8] are designed to expose this information by capturing the flow of data through transformations in data pipelines. Commercial software such as Collibra [2] captures dependencies between databases, tables, and attributes, enabling users to trace the origin and flow of data. While these lineage graphs can reveal where a metric's underlying data originates, the transformation logic often remains cryptic in SQL models or ETL scripts. As a result, understanding how a metric is computed still requires inspecting source code, which requires special expertise and are often cumbersome even for experts. Moreover, existing lineage systems focus on dataset-level lineage rather than explaining the semantics of data transformations that encode business logic. We proceed to highlight the challenges in understanding metric lineage in Example 1.

EXAMPLE 1. *Sheldon, a new data analyst at an e-commerce company, wants to increase Predicted Life-Time Value (`PLTV`), a key metric estimating the total revenue a customer generates over their lifetime. However, he is unfamiliar with `PLTV` and only has access to a dashboard that reports it, without explaining how it is computed. To answer "how to increase `PLTV`?", he must first understand its computation process.*

He inspects the source SQL defining the transformation pipeline to compute `PLTV` as shown in Figure 1 (left). He finds that T100 computes `PLTV` using three intermediate submetrics: `prob_active`, `expected_orders`, and `expected_order_value`. He then traces each metric backward. For `expected_order_value`, he finds a chain of transformations (T20, T21, T22, T50, T97, T98, T99): T50 derives `#_orders`, while T20–T22 and T97–T98 derive `#_total_gross_value`, which are combined in T99. Sheldon also manually traces `prob_active` and `expected_orders` in a similar process.

After several hours, Sheldon concludes that `PLTV` computation involves five submetrics arranged as an approximate 3-ary tree of depth 5, as shown in Figure 1 (right). The tree contains about 120 *nodes, each representing a submetric or data tuple, and* 81 *root-to-leaf paths. Understanding `PLTV` thus requires reasoning over this large dependency tree, which is extremely cumbersome for humans.*

Example 1 highlights the difficulty of understanding how metrics are computed, especially when transformation logic is complex and involves many intermediate submetrics. Users must manually trace transformation steps and relate them to the final metric. Tools such as SQLGlot [10] can parse SQL to expose these steps, but parsing alone does not reveal how metrics depend on one another. In this paper, we address the problem of extracting a metric's lineage, explaining how it is derived from intermediate metrics, and how those, in turn, depend on further submetrics and data.

Once a metric is understood, a natural follow-up is a `how-to` question, e.g., "how can Sheldon increase `PLTV`?" An ideal system should suggest feasible adjustments to dependent metrics and

TID	Description
T20	{SELECT * FROM Orders WHERE `order_date` > `last_month`} Recent_Orders
T21	{JOIN Order_Items and Recent_Orders ON `order_id`} Recent_Order_Items
T22	{JOIN Recent_Products and Products ON `product_id`} Recent_Products
…	…
T50	SELECT COUNT(*) AS `#_orders` FROM Recent_Products
…	…
T97	SELECT (`revenue` – `cost`) AS `gross_value` FROM Recent_Products
T98	SELECT SUM(`gross_value`) AS `#_total_gross_value` FROM Recent_Products
T99	SELECT DIV(`#_total_gross_value`, `#_orders`) AS `expected_order_value`
T100	SELECT MULTIPLY(`prob_active`, `expected_orders`, `expected_order_value`) AS `PLTV`

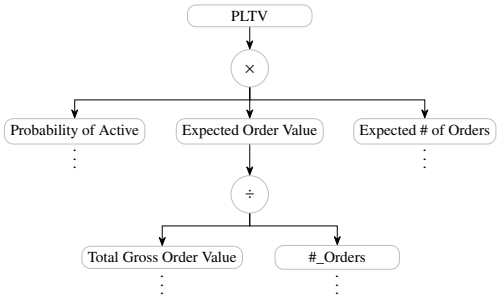

Figure 1: (Left) Transformations required to compute PLTV. (Right) Example hierarchy showing how PLTV is decomposed into sub-metrics.

data values and simulate how these changes propagate to achieve a target metric value. For instance, knowing that `PLTV` depends on `expected_order_value` is not enough to determine how to increase it by adjusting other submetrics or data in a feasible way. The key challenge is understanding how changes in each submetric affect the final metric `PLTV`, as illustrated in Example 2.

EXAMPLE 2. *From the `PLTV` lineage of Figure 1 (right), Sheldon understands that `PLTV` depends on intermediate metrics such as `prob_active`, `expected_orders`, & `expected_order_value`, each derived from further submetrics. Despite this, he still cannot answer "how to increase `PLTV` by 20%?" Multiple strategies exist: increasing a single intermediate metric by 20%, or distributing the increase (e.g., 10% each for `prob_active` and `expected_orders`). However, he cannot easily determine how such changes propagate to `PLTV`, or how to adjust submetrics or data to achieve the target.*

With about 120 *nodes in the computation tree, there are* $2^{120} \approx 10^{36}$ *possible strategies. Accounting for feasible adjustment ranges for each submetric further expands this space. Even if Sheldon could estimate the impact of each strategy, comparing them to identify the most feasible one remains impractical.*

Example 2 illustrates the challenges of answering `how-to` provenance questions on metric lineages. Even when a user understands the metric lineage, determining which submetrics or data to adjust, and by how much, to achieve a desired final metric value requires navigating an exponentially large search space. This motivates the need for a systematic approach that can automatically identify effective adjustment strategies and explain how changes in intermediate submetrics or data values propagate to the final metric.

To support the trust of data-driven decision-making, provenance information must move beyond data-centric lineage and provide accessible, metric-centric explanations that help non-technical users understand how complex metrics are derived.

CLUE. We present CLUE, a system that automatically extracts and visualizes the transformation steps that define a metric, and provides simulation capabilities to support metric adjustment. CLUE captures the transformation logic that defines a metric, and visualizes the metric lineage and the transformation steps in an interactive dashboard. Users can explore the metric lineage and the transformation steps, and see how each step contributes to the final metric. As users explore the transformation steps, they can also see how changes in the intermediate submetrics change the final metric, and get suggestions on how to adjust intermediate metrics and data to achieve a desired value for the final metric. CLUE is built over a SQL-based data pipeline where metrics are defined through multi-layer transformations. Assuming that dashboard metadata and the upstream data pipeline are accessible, CLUE automatically extracts transformation operations and metric dependencies from those sources.

Related Work. Provenance or lineage graphs show the data flow and transformations in various domains such as databases [4, 8] for `how-to` questions, and machine learning pipelines [1, 9]. However, while they often focus on changes in the data and transformations, they do not discuss what these transformations mean in terms of metrics, and how the transformation meanings are related to the final results. Visualization of intermediate transformations [6, 11] allow users to transform and explore data interactively. DataFormulator2 [11] provides an environment for transforming and visualizing intermediate results, while Lux [6] surfaces intermediate insights directly in Jupyter notebooks. In summary, existing work offers little support for helping users understand how transformations produce final metrics, particularly when computations of metrics are complex, multi-step, and involve numerous submetrics that involve complex mathematical and data operations such as joins and aggregations.

Demonstration. In our demonstration, over real world datasets, participants will observe CLUE's capability to decompose complex metrics into interpretable components, reveal how they are computed from underlying data, and support goal-driven exploration by simulating actionable changes across the metric hierarchy. We provide an overview of CLUE in Section 2 and a detailed walkthrough of our demonstration scenario in Section 3.

2 SYSTEM OVERVIEW

CLUE is a metric-centric provenance and simulation system that helps users understand how complex dashboard metrics are derived from intermediate submetrics and how changes in the underlying data affect the final metric value, propagated via the intermediate submetrics. Therefore, the system should be able to (1) extract and represent computation of metrics from transformation pipelines, and (2) generate actionable suggestions for how to achieve a target metric change through data manipulation. However, these requirements pose several challenges: (C_1) the system must handle complex transformation steps into meaningful metric relationships, even though the transformation logic may not explicitly define metrics. (C_2) it must generate feasible suggestions that are relevant to the user's target and grounded in the data in real-time. The possible suggestions are combinatorial in nature, hence the system must efficiently search and rank them to present the most feasible options.

Metric Provenance Graph. CLUE maintains a metric provenance graph for interactive explanation and simulation of complex metrics. The graph is modeled as a bipartite graph consisting of Metric nodes and Rule nodes. Each Metric node connects to a Rule node that encodes how it is computed (e.g., sum, ratio, product, average) from other submetrics. Each Rule node connects to its input Metric nodes via directed edges, representing the computation used to derive its parent Metric node from its children Metric nodes. In Figure 1 (right) `PLTV`, `Probability of Active`, `Expected Order Value`, `Expected # of Orders`, `Total Gross Order Value`, and `#_Orders` are Metric nodes; and Product (×) and Division (÷) are Rule nodes. The `PLTV` node is connected to the Rule node × via a directed edge, which takes input from three Metric nodes, indicating that:

$$\begin{aligned}\texttt{PLTV} = {} & \texttt{Probability of Active} \\ & \times \texttt{Expected Order Value} \\ & \times \texttt{Expected Number of Orders}\end{aligned}$$

Similarly, for the transformation T99 in Figure 1 (left), a division Rule node (÷) connects the input Metric nodes `Total Gross Order Value` and `#_Orders` and maps the result of their division to the output parent Metric node `Expected Order Value`.

Data Transformation Graph. To capture implicit metric derivations from data-level transformation (addressing C_1) and enable actionable data-grounded suggestions (addressing C_2), CLUE also constructs a data transformation graph underneath the metric provenance graph from SQL-based pipelines. We leverage a SQL parsing framework, SQLGlot [10], to convert each query into an abstract syntax tree (AST), from which we identify source tables, intermediate operations, and output columns. Column-level lineages are required for tracing how each output column is derived from upstream columns across common table expressions (CTEs), subqueries, and joins. The resulting graph models transformations as a directed graph, where nodes represent columns and operations, and edges encode "derived-from" relationships with operation types such as aggregation, arithmetic computation, filtering, and joins.

Within this graph, a Column node is considered as a metric or submetric candidate if (i) it is numeric and (ii) it is derived through aggregation or arithmetic expressions. For instance, `#_orders` (from a COUNT aggregation in T50), `gross_value` (from SELECT(revenue – cost) in T97), and `expected_order_value` (from T99) are valid candidates. In contrast, raw attributes (e.g., order_id, subtotal) or derived boolean flags are not valid.

Connecting the Two Graphs. When a metric is derived from a concrete data-table operation (e.g., SUM), CLUE connects the corresponding Metric node in the metric provenance graph to a Column node in the data transformation graph through a BACKED_BY edge. For example, the Metric node `Total Gross Order Value` (T98) is associated with the Column node `gross_value`, which is derived from the SQL expression `SELECT (revenue - cost)` in T97. Through this linkage, CLUE enables users to move seamlessly from high-level metrics to the underlying data transformations that produce them. The unified graph, including both the data provenance and the metric graph, is serialized in JSON format and imported into a Neo4j graph database for efficient querying.

Generating Data Manipulation Suggestions. Given a user-specified target change for a metric, based on the metric provenance graph, CLUE generates data manipulation suggestions to help users reason how changes in the underlying data can achieve the desired metric outcome. To achieve this, CLUE traverses the dependency structure of the target Metric node to identify upstream submetrics and values on the leaf nodes. Next, it generates suggestions from a set of adjustment strategies, such as changing the value of a single component, distributing changes across multiple components, increasing a numerator or denominator during computation of a ratio, or shifting an average by including/excluding high/low value records by changing filter conditions. During this step, CLUE utilizes information about reasonable adjustments from historical data, which guides it to avoid generating suggestions that involve unrealistic adjustments.

For each adjustment strategy, CLUE instantiates candidate suggestions by solving for the leaf-level changes required to move the selected metric toward the user's target. Continuing the example of the `PLTV` metric, assuming the current value of `PLTV` is 10 and the target is 12, CLUE derives a candidate suggestion that increases `Total Gross Order Value` by 200 since this raises `Expected Order Value` from 25 to 30 and yields the desired increase in `PLTV`.

Ranking Data Manipulation Suggestions. CLUE ranks the feasible suggestion candidates using the following criteria: (i) *minimal target overshoot*, favoring candidates whose simulated outcome meets the user-specified target with less overshoot; (ii) *plausibility*, measured by whether the suggested change of values remain within semantically valid bounds, such as non-negative counts, probabilities bounded in $[0, 1]$, or value ranges estimated from percentiles or $mean \pm 2\sigma$; and (iii) *interpretability*, modeled as a proxy for how easy a suggestion is for users to mentally follow through the metric graph. Inspired by prior work on explanation complexity and human-comprehensibility [5], CLUE favors suggestions that modify fewer leaf nodes, propagate through fewer intermediate nodes, and avoid reuse of the same changed node across multiple paths of the metric graph. Finally, CLUE suggests the top-k highest-ranked candidates.

Incremental Simulation Engine. Once a user selects a suggestion, CLUE invokes a simulation engine to visually apply the selected suggestion to show the update in the metric value. It starts by updating the affected leaf values, subsequently traversing the dependency graph upward to identify all ancestor nodes that depend on the modified leaves, and recomputes these nodes in topological order. This incremental recomputation strategy avoids reevaluating the full graph and keeps the interaction responsive. In the running example, selecting the suggestion "Increase `Total Gross Order Value` by 200" updates the leaf `Total Gross Order Value` from 1000 to 1200, recomputes `Expected Order Value` $= 1200/40 = 30$, and then recomputes `PLTV` $= 0.2 \times 30 \times 2 = 12$.

Natrual-Language Explanation Generator. To make CLUE further accessible, especially to non-technical users, we provide natural-language descriptions for both provenance and simulation results using GPT-5.4. For provenance, it states how a metric is composed from submetrics. For simulation, it summarizes the old and new values of the affected nodes, and describes the propagation path of the change through the graph. For example: "Increasing `Total Gross Order Value` by 200 raises `Expected Order Value` from 25 to 30, which increases `PLTV` from 10 to 12."

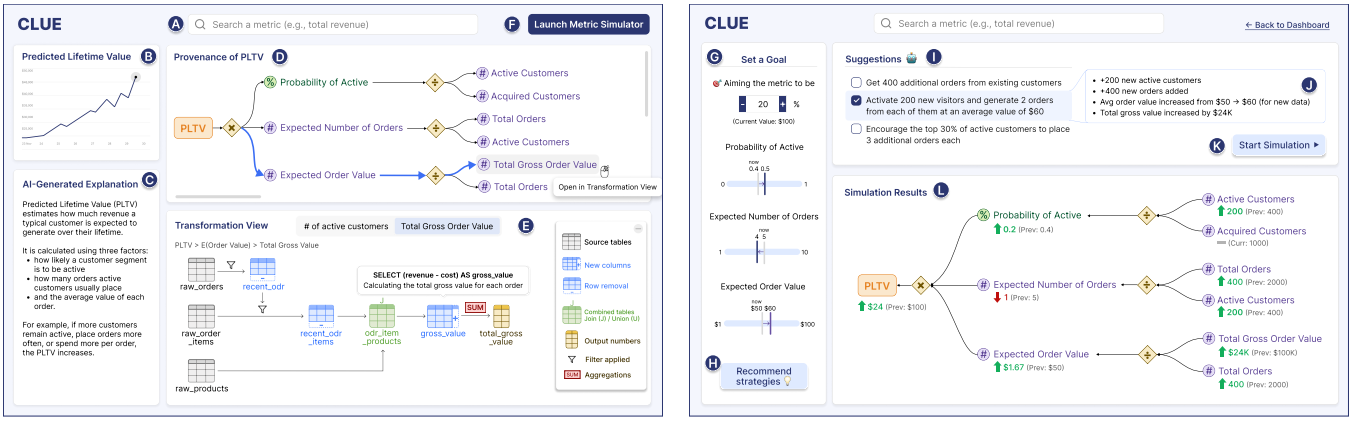

Figure 2: The CLUE interface: (A) search a metric, (B) CLUE displays the metric trend, (C) AI-generated explanation, (D) CLUE visualizes metric lineages, (E) CLUE visualizes data transformation process, (F) launch the Metric Simulator, (G) the user sets up a target value for the metric and tunes the desired values for first-level submetrics, (H) the user requests to generate suggestions to achieve the goal, (I) the user selects a suggestion, (J) the user views the data adjustment summary, (J) start simulation, (K) CLUE visualizes the propagation of value changes for each metric.

3 DEMONSTRATION SCENARIO

We will demonstrate CLUE over an open-source dbt project, Jaffle Shop [3], which contains six tables: customers, orders, order items, products, supplies, and stores. The demonstration guides users through 11 steps (annotated in Figure 2) impersonating Sheldon, who wants figure out how to increase the metric `PLTV` by 20%.

Steps (A), (B), & (C) (Searching for a metric and viewing the description) Sheldon searches for the metric `PLTV` in (A). CLUE displays a snapshot of the metric in (B) along with an AI-generated description explaining how the metric is derived in (C).

Step (D) (Viewing metric provenance) CLUE presents the metric's lineage as a tree. Each child node represents a submetric of the parent and is marked with either # or % so users can distinguish their types. Mathematical operations between sibling nodes are annotated with a yellow diamond on the connecting branches. For example, the root node and the nodes at level 1 represent the equation `PLTV` = `Probability of Active` × `Expected Number of Orders` × `Expected Order Value`. To avoid overwhelming users, only the first-level submetrics are displayed by default. Users can click on a node to expand the next level.

Step (E) (Exploring data provenance) Sheldon clicks "*Open in Transformation View*" for the leaf node `Total Gross Order Value`. CLUE extracts and visualizes the data flow and transformation steps as a DAG in (E). Sheldon can view in-line code context directly within the lineage graph by hovering over the symbol representing each operation and explore transformations across multiple paths by switching between tabs.

Steps (F) & (G) (Simulating metric changes) Sheldon wants to figure out how to increase `PLTV` by 20%. He clicks "*Launch Metric Simulator*" in (F) and sets the target to 20% in (G). CLUE then initializes sliders for the first-level submetrics. When Sheldon adjusts any slider here, CLUE automatically adjusts the other sliders to maintain the target value for `PLTV`.

Steps (H), (I) & (J) (Requesting suggestions) Sheldon clicks "*Recommend strategies*" in (H), and CLUE suggests several approaches to achieve the goal set in (G). The data change summary of an option will show in (J) when the user selects it.

Step (K) & (L) (Simulating a suggestion) Sheldon selects the second suggestion and clicks "*Start simulation*" in (K). CLUE then computes the updated values for each metric and displays the result in (K). Changes are highlighted in green or red to indicate increases or decreases in metric values.

Demonstration engagement. Our target users include non-technical decision makers and data professionals who need to understand how a complex metric is derived. After the guided demonstration, participants can use their own datasets to explore the relationships between a metric and its components, and observe how changes in the underlying data propagate to the metric value. The key takeaway from our demonstration is that CLUE can support data-driven decision-making via metric-centric explanations to help users understand how complex metrics are derived and how to adjust data to achieve the desired change in those metrics.

REFERENCES

[1] A. Chapman, L. Lauro, P. Missier, and R. Torlone. 2024. Supporting Better Insights of Data Science Pipelines with Fine-grained Provenance. *ACM Trans. Database Syst.* 49, 2 (2024), 6:1–6:42.
[2] Collibra. 2020. Data Lineage Tool. `www.collibra.com`
[3] dbt labs. 2023. Jaffle-Shop. `github.com/dbt-labs/jaffle-shop`
[4] S. Galhotra, A. Gilad, S. Roy, and B. Salimi. 2022. Hyper: Hypothetical reasoning with what-if and how-to queries using a probabilistic causal approach. In *SIGMOD*. 1598–1611.
[5] I. Lage, E. Chen, J. He, M. Narayanan, B. Kim, S. J. Gershman, and F. Doshi-Velez. 2019. Human evaluation of models built for interpretability. In *AAAI*, Vol. 7. 59–67.
[6] D. J. L. Lee, D. Tang, K. Agarwal, T. Boonmark, C. Chen, J. Kang, U. Mukhopadhyay, J. Song, M. Yong, M. A. Hearst, and A. G. Parameswaran. 2021. Lux: Always-on Visualization Recommendations for Exploratory Dataframe Workflows. *PVLDB* 15, 3 (2021), 727–738.
[7] R. Matheus, M. Janssen, and D. Maheshwari. 2020. Data science empowering the public: Data-driven dashboards for transparent and accountable decision-making in smart cities. *Government Information Quarterly* 37, 3 (2020), 101284.
[8] A. Meliou and D. Suciu. 2012. Tiresias: the database oracle for how-to queries. In *SIGMOD*. ACM, 337–348.
[9] D. B. Pina, A. Chapman, L. N. O. Kunstmann, D. de Oliveira, and M. Mattoso. 2024. DLProv: A Data-Centric Support for Deep Learning Workflow Analyses. In *DEEM@SIGMOD*. ACM, 77–85.
[10] Tobymao. 2023. SQLGlot. `https://github.com/tobymao/sqlglot`
[11] C. Wang, B. Lee, S. M. Drucker, D. Marshall, and J. Gao. 2025. Data Formulator 2: Iterative Creation of Data Visualizations, with AI Transforming Data Along the Way. In *CHI*. ACM, 677:1–677:17.