Nexa: Automatically Surfacing Business Impacting Insights in E-commerce Applications

Smart Sun[1], Sayan Sinha[1,2], Haijie Wu[1], Joel Goldfoot[1], Aditya Ganjam[1], Jibin Zhan[1], Evan Chan[1], Qichu Gong[1], Vipul Harsh[1], Bo Lin[1], Pawas Ranjan[1], Wei Wang[1], Zhan Yang[1], Ningning Hu[1], B. Aditya Prakash[1,2], Vyas Sekar[1,3], Hui Zhang[1,3]
[1]Conviva, [2]Georgia Tech, [3]Carnegie Mellon

Abstract

Internet-scale e-commerce storefronts serve millions of users (and increasingly user appointed agents). These storefronts are being rearchitected as compound AI systems with agentic workflows for customer interactions and backend processing. As this AI transformation and agentic economy is underway, product teams need to get actionable insights into business-impacting outcomes. Classical approaches such as static funnels or static dashboards cannot deal with the scale, diversity, and contextual interactions that happen over billions of user interactions. As such, we need novel agentic approaches to automatically surface business-impacting insights. We present Nexa, an agentic framework that surfaces business insights automatically. We formalize the target of automated insight discovery in terms of Contrastive Stateful Trajectories (*CST*): a structural specification over contextual and sequential behavioral patterns whose presence or absence significantly shifts a business KPI across user cohorts. Nexa satisfies three design requirements simultaneously: **expressivity** through the *CST* abstraction, **scalability** through a custom analytics backend for *CST* computations, and **explainability** by overlaying usable presentation layers for analysts to verify the insights. We demonstrate Nexa on representative workloads and show that it surfaces actionable contextual patterns spanning user, app, agent, and backend behaviors.

CCS Concepts

• **Networks** → **Network monitoring**; *Network reliability*; *Network management*.

Keywords

Internet services, Network Telemetry, Pattern Analytics

ACM Reference Format:
Smart Sun[1], Sayan Sinha[1,2], Haijie Wu[1], Joel Goldfoot[1], Aditya Ganjam[1], Jibin Zhan[1], Evan Chan[1], Qichu Gong[1], Vipul Harsh[1], Bo Lin[1], Pawas Ranjan[1], Wei Wang[1], Zhan Yang[1], Ningning Hu[1], B. Aditya Prakash[1,2], Vyas Sekar[1,3], Hui Zhang[1,3], [1]Conviva, [2]Georgia Tech, [3]Carnegie Mellon, . 2026. Nexa: Automatically Surfacing Business Impacting Insights in E-commerce Applications. In *ACM Conference on AI and Agentic Systems (CAIS '26), May 26–29, 2026, San Jose, CA, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3786335.3813185

CAIS '26, San Jose, CA, USA

ACM ISBN 979-8-4007-2415-2/2026/05
https://doi.org/10.1145/3786335.3813185

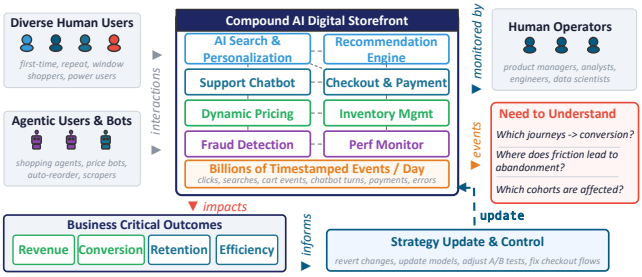


Figure 1: A modern digital storefront comprises multiple AI-driven components serving diverse human and agentic users, generating billions of timestamped events daily. Human operators must understand user behavioral patterns across this ecosystem to improve business-critical outcomes.

1 Introduction

Digital storefronts are evolving from monolithic applications into compound AI systems (Figure 1). A modern storefront weaves together AI-powered search, recommendation engines, support chatbots, and backend agents for pricing, refund processing, and fraud detection. Both human shoppers and autonomous clients interact with these components, together generating billions of events per day [17].

As user journeys now span multiple AI (e.g., conversational assistants) and non-AI (e.g., product pages) components, an interaction in one aspect of the system can have cascading implications for another. For instance, a recommendation agent and a promotion agent may issue contradicting suggestions, eroding user trust. Coupled with the diversity of user behaviors, this makes the behavioral space combinatorially large. As such, the manual investigation and classical product analytics tools and capabilities (e.g., static funnels) that operators conventionally rely on today have fundamental gaps.

Given the scale, complexity and diversity of user behaviors, there is a critical need for an *agentic system* to identify opportunities to improve customer experience and business outcomes. Designing such an agentic system for surfacing insights for next-generation e-commerce applications raises fundamental algorithmic, system design, and usability questions:

- *What is the right* **abstraction** *to formalize such critical insights?* Today, product analytics tools rely on pre-defined aggregates, attributes, and static funnels [3, 12]. These are limited and will not be able to capture the diverse dynamic interactions in future compound AI systems. Other approaches look at backend metrics and traces, but fail to connect them to outcomes [11, 13, 15].

- *Can we* **automatically** *identify insights* **at scale***?*
  Classical data processing solutions already struggle with conventional relational analytics at scale [2, 9]. We may need even more expressive analytics at scale that pushes the boundary of conventional analysis.
- *How do we present these to human decision makers in a way that is meaningful and* **explainable***?*
  Since these insights drive business decisions at scale, they must be explainable and usable to the analyst.

We present Nexa, an agentic framework that automatically surfaces business-critical insights for e-commerce storefronts. Nexa takes in available interaction data (e.g., client logs, app logs, agent interaction traces, backend traces and agent-agent interaction logs). Given a high-level intent from the analyst Nexa suggests why business KPIs may be impacted and how these can be potentially remediated.

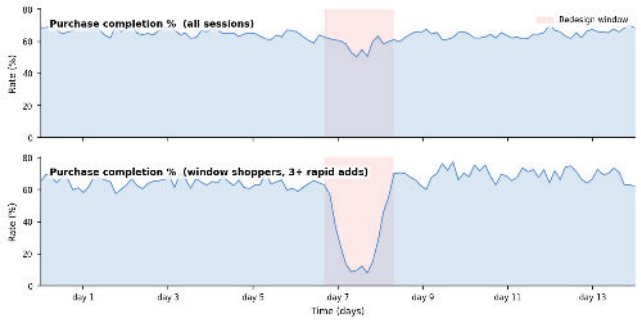
Nexa introduces a novel abstraction for capturing business-critical insights we call *Contrastive Stateful Trajectories* (*CST*). First, we formalize behavioral patterns as *stateful trajectories*. A *state* is a business-meaningful condition computed from raw events, such as a user's cart value tier or login frequency. Second, we argue that an actionable behavioral insight is inherently *contrastive*: an isolated metric, like "5% of users reach checkout", is rarely useful. A *contrastive* version of the same question is far more actionable: "users who applied a coupon reach checkout at only 2%, vs. 8% for those who did not." What matters is that a particular group of users exhibit a behavioral pattern differently from others, and that the difference is large enough to act on. Taken together, a stateful trajectory is *contrastive* if its effect on a business KPI varies significantly across user segments defined by a derived state. The shift is not uniform, but concentrated in particular populations.

We implement a custom DSL to express hypotheses using the *CST* abstraction and a novel LLM-assisted approach for exploring the space of candidate hypotheses efficiently. We also implement a scalable vectorized backend capable of analyzing large volumes of data in context, including client-side events, app events, agent-user interaction traces, multi-agent traces, and tool calls. We demonstrate the utility Nexa on representative workloads across ecommerce, streaming, and chatbot domains, showing that it surfaces high-signal *CST*s at interactive latency.

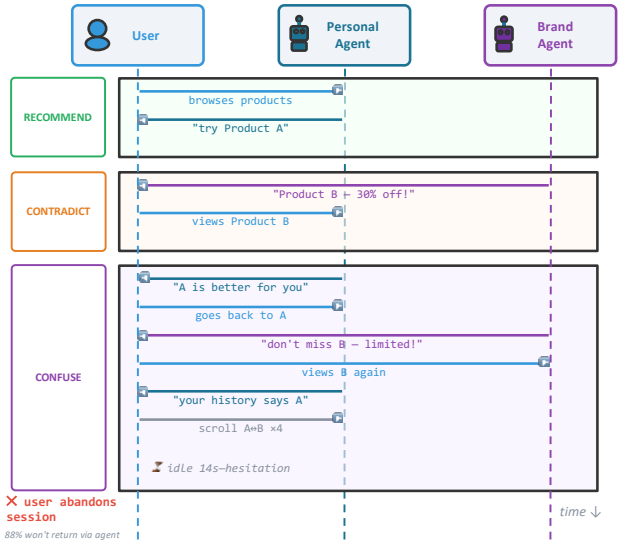
2 Motivating Examples

We present two examples inspired by real-world incidents to illustrate why conventional analytics solutions (e.g., flat aggregates, static funnels) are not sufficient.

Example 1: Checkout conversion: An e-commerce platform redesigns its checkout flow. Within hours, the overall purchase completion rate drops from ~68% to ~60% (Figure 2a, top panel). The team notices the decline and reverts the change. While the immediate issue is resolved, the root cause remains unclear: the aggregate tells us conversions fell, not *why*. In this case, we find that *casual shoppers*, i.e., users with fewer than two purchases in the past month, who added three or more items to cart in rapid succession completed at only ~12% (Figure 2a, bottom panel). Committed shoppers with the same rapid-add pattern, and gradual window shoppers, were unaffected as they push through checkout friction.



(a) Demonstrating a checkout redesign failure. The trajectory T = `(window, rapid_add)` → `(window, rapid_add)` → `(window, rapid_add)` isolates window shoppers who add items rapidly. During the redesign, the overall purchase completion rate falls modestly from ~68% to ~60% (top), but the affected cohort drops from ~68% to ~12% (bottom).



(b) The trajectory T = `recommend` → `contradict` → `confuse` → `abandon` captures sessions in which an external shopping agent and the storefront's agents give conflicting suggestions. Among sessions matching T, first-time shoppers abandon at 68%, whereas repeat shoppers abandon at 14%.

Figure 2: This figure presents two examples of *CST*s. An aggregate KPI indicates that something is wrong, but the failure mode becomes clear only after isolating a trajectory and comparing its effect across cohorts.

The redesign introduced a confirmation step causing friction and uncommitted buyers left.

Example 2: Conflicting Agent Recommendations: The overall conversion rate holds steady, raising no alerts, but the analyst noticed that first-time shoppers do not return. In this case, we find a four-step pattern (Figure 2b). The user arrives with a product recommendation from an external AI assistant (`recommend`); the storefront's agents redirect to a different product (`contradict`); the user begins navigating back and forth between the two options

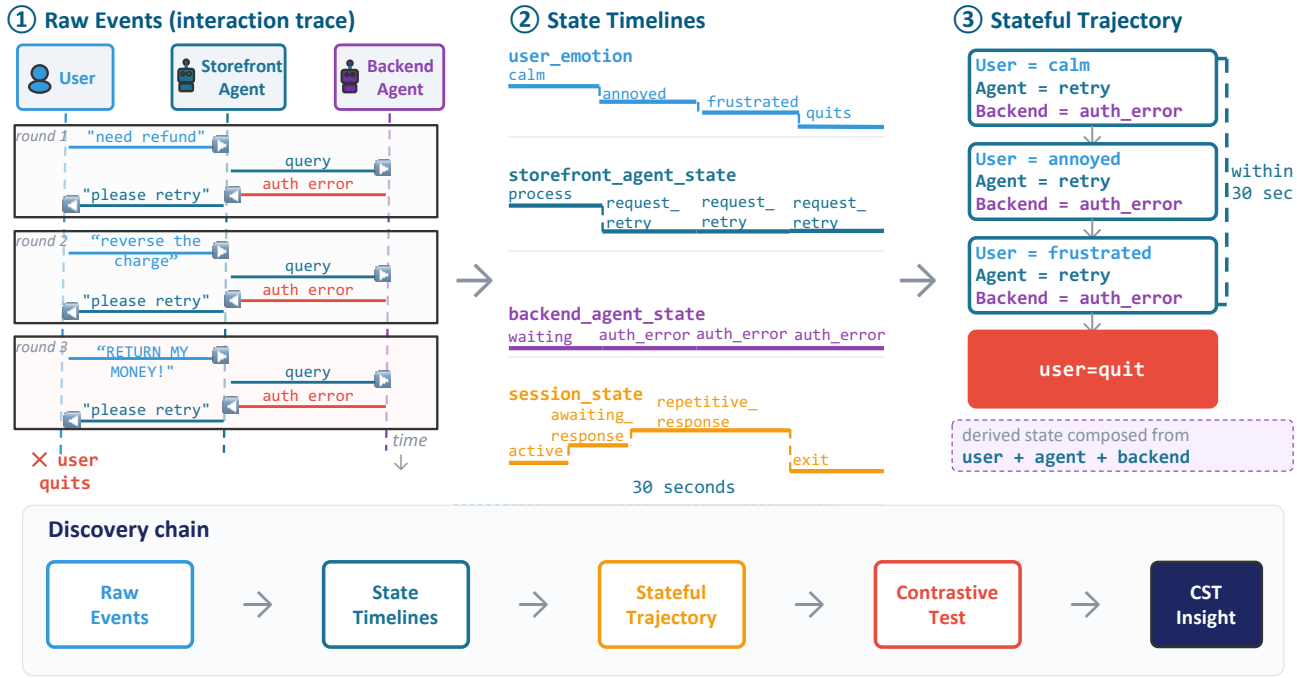


Figure 3: Parallel state timelines for a customer-support session. Three derived states, `user_emotion`, `storefront_agent_state`, and `backend_agent_state`, evolve independently over time. A trajectory T captures three consecutive composite steps: `(calm, retry, auth_error)` → `(annoyed, retry, auth_error)` → `(frustrated, retry, auth_error)`. Neither key alone identifies this pattern; composing them into a single trajectory node is what isolates a specific interaction. This illustrates how trajectories over multiple state dimensions go beyond what any single event field can express.

(`confuse`); and the user abandons the session (`abandon`). The trajectory T matches 8,741 sessions over seven days. Among first-time shoppers, 68% of sessions matching T end in abandonment, with conversion dropping to 1.8% versus 9.2% for non-conflicted sessions. Among repeat shoppers, only 14% abandon; among loyalty members, 6%. First-time shoppers have not built enough trust to tolerate inter-agent contradiction, while repeat and loyalty users push through it.

Both examples highlight the need for an *expressive* system to surface such insights. It also highlights limitations of conventional approaches that rely on static user attributes, pre-configured funnels and transitions, low-dimensional reasoning, or failing to connect to the outcome impact. First, the actionable signal lies not in any single static attribute that manifests in a single entity, but in the composition of multiple and derived states. For instance, in Example 1, we have to model the stateful behavior of the rate of session-level cart addition combined with derived attributes by looking at user-level purchase history. Similarly, in Example 2, we have to look at the content to identify semantic disconnects between agents and combine with user-level purchase behavior. Second, we need to tie the behavior to the business impacting user behaviors; just looking at backend logs or agent conversations may not reveal the business impacting outcome. Finally, these are high dimensional and nuanced behaviors that need to be unearthed from trillions of raw events; doing it manually is impossible and we need an automated *scalable* approach, but *explainable* to the human decision maker.

3 Design

Nexa takes a natural-language KPI question (e.g., "why did checkout conversion drop?") and uses an agentic workflow over interaction events, logs, and traces to surface actionable insights with explainable evidence trails. We first formalize Contrastive Stateful Trajectories (*CST*), the foundational abstraction (§3.1), then describe the end-to-end workflow (§3.2).

3.1 *CST* Abstraction

In a compound AI system, entities such as users, AI agents, and backend services emit timestamped *events*. An event is a single recorded occurrence, such as a page view, add-to-cart action, model invocation, or backend error, together with its associated attributes. We organize these events into *sessions* so that analysis is anchored to a bounded interaction, such as a shopping visit from landing page to checkout or exit. From this event stream, the system constructs *state timelines* for business-relevant attributes. For a given attribute key—for example, current page, buffer state, network quality, or CPU usage—the timeline records how the attribute evolves over time. Figure 3 illustrates such evolutions; for instance, `buffer_state` and `user_action` are both *base* states, read directly from a single event field. Other states are *derived*: functions on multiple signals that capture business-meaningful conditions no single field records, such as whether a user is a returning visitor or whether a chatbot interaction has turned frustrated.

A *stateful trajectory* is an ordered sequence of states or transitions over one or more of these timelines, together with optional temporal constraints, wildcard steps, per-step attribute filters, and containment predicates. This gives the system a vocabulary for expressing behaviors at the level analysts actually care about, rather than as isolated event filters. We call a trajectory T a *Contrastive Stateful Trajectories* (*CST*) when its presence materially shifts a business KPI relative to its absence:

$$\text{Dist}(\text{KPI} \mid T \text{ present}) - \text{Dist}(\text{KPI} \mid T \text{ absent}) > \delta. \quad (1)$$

A *CST* captures both structure and contrast: the trajectory's occurrence is what moves the KPI. The agent further decomposes this effect across cohorts defined by state values to identify which user populations are most affected. Surfacing *CST*s is the target output of the system.

Using this abstraction, hypotheses to explain business-impacting outcomes are expressed in a domain-specific language (DSL) that names trajectory structure, derived states, and cohort dimensions. An LLM-based orchestrator frames the analyst question into candidate trajectories, tests them against the analytics backend, and iteratively refines the search until a contrastive signal emerges. The final output is a summary of the confirmed *CST*s together with supporting evidence.

3.2 Workflow

The orchestrator proceeds through four phases, as described below: hypothesis generation, evaluation, presentation, and refinement.

Hypothesis generation. The planner LLM retrieves known derived state and event filter definitions from the catalog, falling back to raw schema discovery when the catalog is insufficient. When existing attributes do not cover the hypothesis, the planner generates new derived state definitions on demand from natural-language descriptions. From these building blocks the planner proposes candidate stateful trajectories in the DSL.

Scalable evaluation. A separate query-generator LLM translates each candidate trajectory into valid DSL query syntax, thereby separating *what to explore* (planner) from *how to query* (generator) and keeping each LLM call focused and reducing hallucinated queries. The resulting query is submitted to a vectorized analytics backend: a stateless engine that evaluates the trajectory in a single pass over columnar event data, matching steps and propagating attribute values without materializing intermediate tables. The backend scales horizontally because statelessness lets each query execute independently, with no coordination between instances.

Explainable presentation. A candidate trajectory whose KPI delta exceeds the threshold δ is promoted to a confirmed *CST*. The confirmed *CST*s, their cohort-level KPI deltas, and the full execution graph (mapping each hypothesis to the corresponding queries, the results returned, and the planner decisions) are presented to the analyst so they can audit and replay the reasoning path.

Iterative refinement. If no trajectory exceeds δ, the planner automatically refines its search for up to five iterations, extendable on the analyst's request; the analyst may optionally seed any iteration with a natural-language hint. When catalog-defined constructs prove insufficient, the system can generate new side-effect-free tools to extend the planner's analytical reach; the refined hypothesis is then fed back to the evaluation step.

Figure 4: Nexa conceptual user interface: user query, generated analysis pipeline, results, and *CST* visualizations.

4 Demonstration

Figure 4 shows a proof-of-concept Nexa user interface. A demonstration video can be viewed at this link. We use this conceptual prototype to illustrate how Nexa satisfies the three design requirements. Nexa enables an analyst to pose a natural-language question about a degraded KPI. Then, it constructs derived states from raw event sequences and generates a dataflow pipeline, demonstrating expressivity through the *CST* abstraction. Every pipeline node is inspectable with intermediate data and generated Python code (Figure 4), providing explainability. Caching layers and vectorized backend enable interactive latency, addressing scalability. The output presents the relevant *CST*s and visualizations of state trajectories.

5 Related Work

Ad hoc analytics and root cause localization. Platforms such as Amplitude [3], Heap [12], and observability tools [11, 13, 15] let analysts write custom queries and inspect dashboards but do not scale to multi-step trajectories over derived states. Root cause methods such as Adtributor [6], Scorpion [19], emerging patterns [8], and related contrast mining methods [5] localize KPI shifts to flat attribute combinations but cannot express temporal sequences of composed states.

Data analysis agents and pattern mining. LLM-based data analysis agents, including NL2SQL systems [7, 18, 20, 21], map each question to a single query or code block, with no iterative hypothesis exploration, contrastive framing, or domain-specific abstraction for compound AI applications [1]. Sequential pattern mining [16] finds frequent subsequences over discrete event names but produces no contrastive signal and lacks temporal constraints or state composition. Complex event processing engines [4, 9] match temporal event patterns but lack derived state composition and contrastive KPI analysis.

Nexa builds on the vision of automatically surfacing improvement opportunities via derived attributes [10] and the timeline abstraction for time-state analytics [14], combining the contrastive framing of root cause localization with the temporal structure of pattern languages, driven by an LLM exploration loop over a scalable analytics backend.

References

[1] 2024. The Shift from Models to Compound AI Systems. https://bair.berkeley.edu/blog/2024/02/18/compound-ai-systems/.

[2] Tyler Akidau et al. 2015. The Dataflow Model: A Practical Approach to Balancing Correctness, Latency, and Cost in Massive-Scale, Unbounded, Out-of-Order Data Processing. *Proc. VLDB Endow.* 8, 12 (2015), 1792–1803. doi:10.14778/2824032.2824076

[3] Amplitude, Inc. 2025. Amplitude Analytics. https://amplitude.com Accessed 2026-03-08.

[4] Apache Flink Project. 2026. FlinkCEP — Complex event processing for Flink. https://nightlies.apache.org/flink/flink-docs-stable/docs/libs/cep/ Accessed 2026-02-23.

[5] Stephen D. Bay and Michael J. Pazzani. 2001. Detecting Group Differences: Mining Contrast Sets. *Data Mining and Knowledge Discovery* 5, 3 (2001), 213–246. doi:10.1023/A:1011429418057

[6] Ranjita Bhagwan, Rahul Kumar, Ramachandran Ramjee, George Varghese, Surjyakanta Mohapatra, Hemanth Manoharan, and Piyush Shah. 2014. Adtributor: Revenue Debugging in Advertising Systems. In *Proceedings of the 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. 43–55.

[7] Asim Biswal, Liana Patel, Siddarth Jha, Amog Kamsetty, Shu Liu, Joseph E Gonzalez, Carlos Guestrin, and Matei Zaharia. 2024. Text2sql is not enough: Unifying AI and databases with TAG. *arXiv preprint arXiv:2408.14717* (2024).

[8] Guozhu Dong and Jinyan Li. 1999. Efficient Mining of Emerging Patterns: Discovering Trends and Differences. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 43–52. doi:10.1145/312129.312191

[9] Opher Etzion et al. 2006. Event-Driven Architectures and Complex Event Processing. In *2006 IEEE International Conference on Services Computing (SCC'06)*. IEEE, Washington, DC, USA. doi:10.1109/SCC.2006.49

[10] Vipul Harsh, Sayan Sinha, Henry Milner, Haijie Wu, B Aditya Prakash, Vyas Sekar, and Hui Zhang. 2025. Automatically Surfacing Opportunities for Improvements In Internet-Scale Applications. In *The 24th ACM Workshop on Hot Topics in Networks (HotNets '25)*. doi:10.1145/3772356.3772423

[11] Vipul Harsh, Wenxuan Zhou, Sachin Ashok, Radhika Niranjan Mysore, Brighten Godfrey, and Sujata Banerjee. 2023. Murphy: Performance diagnosis of distributed cloud applications. In *Proceedings of the ACM SIGCOMM 2023 Conference*. Association for Computing Machinery, New York, NY, USA, 438–451. doi:10.1145/3603269.3604877

[12] Heap, Inc. 2025. Heap. https://heap.io Accessed 2026-03-08.

[13] Jonathan Mace, Ryan Roelke, and Rodrigo Fonseca. 2017. Pivot tracing: Dynamic causal monitoring for distributed systems. *ACM Transactions on Computer Systems (TOCS)* 35, 4 (2017), 1–28. doi:10.1145/3208104

[14] Henry Milner, Yihua Cheng, Jibin Zhan, Hui Zhang, Vyas Sekar, Junchen Jiang, and Ion Stoica. 2023. Raising the Level of Abstraction for Time-State Analytics With the Timeline Framework. In *CIDR*.

[15] OpenTelemetry. 2026. Observability Primer: Introduction to Observability. https://opentelemetry.io/docs/concepts/observability-primer/ Accessed 2026-03-08.

[16] Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Meichun Hsu. 2001. PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth. In *Proceedings of the 17th International Conference on Data Engineering (ICDE)*. 215–224.

[17] David M. Rothschild, Markus Mobius, Jake M. Hofman, Eleanor W. Dillon, Daniel G. Goldstein, Nicole Immorlica, Sonia Jaffe, Brendan Lucier, Aleksandrs Slivkins, and Matthew Vogel. 2025. The Agentic Economy. arXiv:2505.15799 [cs.CY] https://arxiv.org/abs/2505.15799

[18] Sinaptik-AI. 2024. PandasAI: Conversational Data Analysis. https://github.com/Sinaptik-AI/pandas-ai Accessed 2026-03-08.

[19] Eugene Wu and Samuel Madden. 2013. Scorpion: Explaining Away Outliers in Aggregate Queries. *Proceedings of the VLDB Endowment* 6, 8 (2013), 553–564. doi:10.14778/2536354.2536356

[20] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. In *The Eleventh International Conference on Learning Representations*. https://openreview.net/forum?id=WE_vluYUL-X

[21] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 3911–3921. doi:10.18653/v1/D18-1425