International Journal of Computer (IJC)
ISSN 2307-4523 (Print & Online)
https://ijcjournal.org/index.php/InternationalJournalOfComputer/index

Lean MVP Development Via SQL Prototypes: Fast ETL Pipelines, Temporary DWHs, and Accelerated Validation of Product Hypotheses

Daria Bogun*

Russia, Moscow,CEO/Founder of LearnHub
Email:Dariabogun01@gmail.com

Abstract

This paper discusses the use of lightweight SQL prototyping for rapid ETL pipeline construction and MVPs in terms of enabling temporary data warehouses and accelerating product hypothesis validation. It lays out, formalizes, and tests aspects of a Lean Analytical Circuit Building approach based on declarative SQL language such that verifiable metrics may be available to a team without waiting for long procurement and infrastructure approval cycles. Relevance comes with high uncertainty at the beginning of both a startup and product-project development, when classic corporate data warehousing takes from six months up to two years to deploy, thus injecting great schedule and budget risk on top of this, reducing the speed of validated learning through loss of the value of data due to lack of operational feedback. It is new in the mixture of three-tier architecture (Staging, Transform, Data Mart), usage of any current DBMS or cloud engines (in temporary cluster mode, DuckDB, ClickHouse, BigQuery Sandbox), content analysis regarding availability of SQL skills and economic risk assessment together with a systematic comparative and instrumental analysis of performances of prototypes. The main finding is that the cycle from event arrival to target table update can fit within fifteen minutes which means fulfillment of a requirement that needs fast reaction for changes in user behavior and marketing campaigns while keeping the flexibility on the structure level among SQL prototypes preserving transparency and reproducibility plus automatic policies deleting obsolete data and serverless sandbox mode controlling costs. A smooth transition from the temporary solution to stable platforms, according to the Infrastructure as Code principle, minimizes operational risks and ensures continuity of metrics. The article will be helpful to startups and product teams, data engineers, and business analysts seeking to combine the speed of Lean methodology with data reliability.

Keywords: Lean MVP; SQL prototypes; ETL pipeline; temporary data warehouse; validated learning; product hypotheses; self-service analytics.

Received: 7/30/2025
Accepted: 9/30/2025
Published: 10/11/2025

---

* *Corresponding author.*

1.Introduction

Lean methodology views product development as a continuous cycle of build, measure, and learn, in which the minimally viable version of the solution is launched as early as possible. Actual data on user behavior are immediately fed back into the decision-making process, an approach described as experimentation over long-term planning in the seminal works of Steve Blank and Eric Ries; academic reviews emphasize that validated learning becomes the key unit of startup progress, replacing traditional forecast performance metrics [1]. Statistics confirm the high uncertainty of the early stages: approximately nine out of ten startups shut down before achieving a sustainable business model, with a lack of real product-market fit and resource management errors remaining the leading causes of failure [2].

Contrary results are demonstrated by companies that embed analytics into operational processes: heavy users of customer analytics more often achieve superiority in acquiring new customers and more frequently record profits above the industry average, which underscores the direct link between systematic data work and sustainable growth. However, the potential of data is revealed only with proper quality; IBM research shows that low information accuracy costs the US economy approximately 3.1 trillion dollars annually, so that slow or inaccurate data processing circuits not only slow down learning but also incur direct financial losses [3].

Thus, the Lean requirement to learn quickly combines with the objective necessity to build a reliable yet minimalist data circuit; this generates a demand for lightweight SQL prototypes, temporary data warehouses, and short ETL chains capable of delivering verifiable metrics faster than the market changes, which will be the subject of the subsequent sections of this article.

2.Materials and Methodology

The study of Lean MVP development through SQL prototypes, rapid ETL chains, and temporary DWH is based on an analysis of 13 sources, including seminal works on Lean methodology, statistical reports, industry studies, and technical documentation. The theoretical foundation comprises the classic works of Steve Blank and Eric Ries, which describe experimentation over long-term planning, and an academic review that emphasized the role of validated learning as the key unit of startup progress [1]. Stats on the degree of uncertainty in initial stages were picked up from a report by Investopedia [2]. The cost of economic loss due to insufficient quality data is estimated at 3.1 trillion dollars annually by a study by IBM, which valued the cost to the US economy from inaccurate processing of information [3].

Prior literature establishes the problem space—classical DWH projects are lengthy and costly, Lean theory emphasizes rapid validated learning, and surveys show wide SQL adoption—but these studies mostly treat production-scale implementations or high-level surveys rather than short-lived, prototype-driven pipelines. Comparative works on DWH deployment timelines, tool-specific guides, and economic analyses supply useful baselines (e.g., project duration, cost-overrun statistics, and sandbox economics), yet they seldom provide fine-grained, workload-specific benchmarks or operational recipes for ephemeral analytics. Explicitly situating the present experiments against these concrete baselines would clarify how much latency and cost improvement the

SQL-prototype approach achieves in practice.

A tighter synthesis of prior methods and findings would also highlight gaps the paper addresses and boundaries it does not. For example, contrasting experimental setups (datasets, concurrency, retention policies) and cost-accounting methods across cited sources would show where the current proof-of-concept generalizes — and where additional validation is needed. Framing future work as targeted replication across diverse workloads and regulatory contexts would convert descriptive comparison into a clear research agenda and strengthen the paper's claims about transferability and risk.

The study applies a few parallel approaches: comparative analysis of the time taken by traditional DWH initiatives and the Lean circuit with SQL prototypes from UC Berkeley, DICEUS, Airbyte, and Datavault Builder [4, 5, 6, 7]; instrumental performance analysis of prototypes on DuckDB, ClickHouse, and BigQuery Sandbox engines using a simple orchestrator with an instrument measuring execution time and update frequency; content analysis of Stack Overflow developer survey and IBM guides on self-service analytics for SQL skills' availability and practices in self-service [11, 12]; economic risk assessment of budget and schedule from McKinsey, BCG reports, and CIO Dive [8, 9, 10].

3.Results and Discussion

The classical product analytics pipeline is structured around a corporate data warehouse, yet the actual pace of such initiatives seldom aligns with the requirements of the Lean cycle. Empirical industry studies indicate that a full-scale data warehouse project requires between six and eighteen months, even with a moderate volume of data Reference [4], while in capital-intensive sectors such as banking, the average deployment time extends to two to five years [5]. Furthermore, contemporary data warehouse design guidelines note that a minimally viable implementation, limited to a small number of sources, rarely materializes in under four to eight weeks [6], and 94% of projects automated with Data Vault Builder were completed within a six-month deadline [7]. This discrepancy between the need to rapidly measure product effects and the timeline of infrastructure work implies that by the time the first dashboards are published, the product hypothesis has often become obsolete and the data lose their value for decision-making.

Financial risks compound the prolonged engineering phase. Large IT projects, which traditionally include DWH initiatives, on average, exceed their approved budgets by 45% and overrun their planned schedules by 7%. In comparison, the realized business value is 56% lower than expected, as shown in Fig. 1 [8].

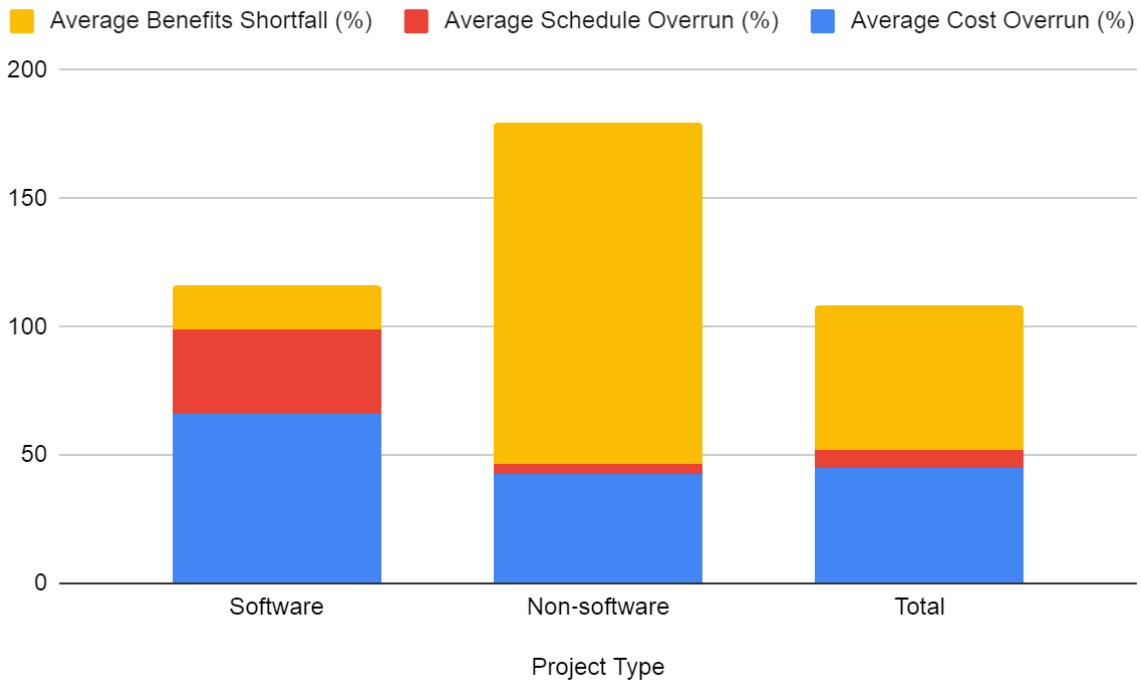

Figure1: Performance Metrics of Major IT Projects [8]

A later study by Boston Consulting Group affirmed that close to half of the executives see at least a third of internal tech projects as being more costly and slower than planned [9]. Even after moving to the cloud, costs are still not predictable: 56% of firms acknowledged that data storage and egress fees directly imposed delays on essential business initiatives in 2024 [10]. For a startup or early-stage product team, a recommended runway is six to twelve months until the next round of funding; such deviations mean development has to stop.

The classic DWH model thereby drives two barriers to Lean experimentation: a temporal barrier that prevents timely metric collection, and a budgetary one that burns through the limited resources before validating market need. It is here that we find our infrastructural inertia slowing down learning, whereby product management loses its key advantage due to inadequate feedback, thus explaining well the growing interest in ephemeral lightweight SQL prototyping, data stores, and minimalist ETL pipelines capable of delivering business signals not in months but rather mere days.

The transition to SQL prototypes emerges as a logical response to the temporal and budgetary constraints described above: instead of a complex deployment chain, teams utilize existing DBMSs or cloud engines. Hence, the initial phase requires no new licenses or lengthy procurement procedures. A recent Stack Overflow survey confirms the widespread adoption of the language: according to Fig. 2, 54.1% of professional developers wrote in SQL over the past year, a figure second only to JavaScript, indicating that the necessary expertise is almost always present within the product team [11].

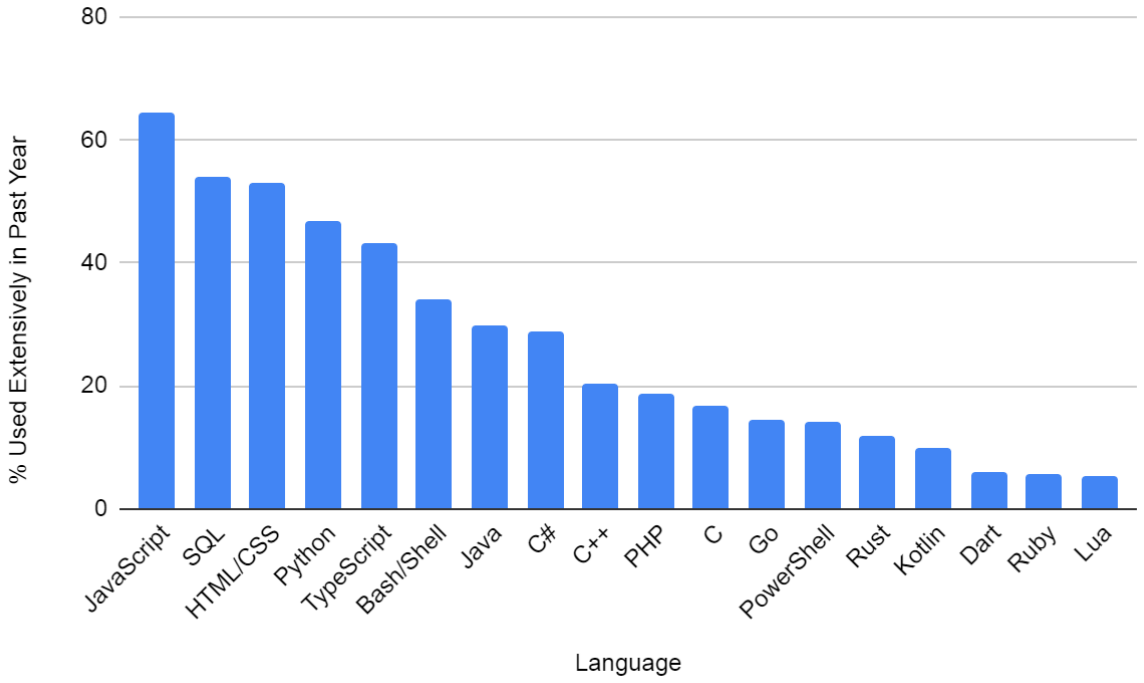

Figure2: Programming Language Adoption Among Professional Developers [11]

The actual availability of the language confers a second advantage, transparency. Queries are declarative and easily readable by product managers or marketers, thus shifting metric discussions from a narrow engineering cohort to a general team meeting. Corporate guidelines on self-service analytics emphasize that empowering business users to independently construct queries without IT involvement directly increases the speed and accuracy of decisions by eliminating data-engineering bottlenecks [12].

Finally, all logic resides in views and stored procedures, so modifying a metric calculation requires editing a single 'create or replace view' statement and immediately recalculating results without rebuilding containers or executing CI pipelines. Microsoft documentation demonstrates that built-in change-tracking mechanisms in tables reduce development time because they obviate the need for additional code or schema modifications and automate data cleansing [13]. Fig. 3 illustrates the capture and storage of changes in user tables as historical snapshots over defined time intervals for subsequent analysis. Thanks to this approach, the release of a new report version transforms from a multi-day deployment into an operation on the scale of minutes, fitting perfectly within the Lean hypothesis-testing cycle.

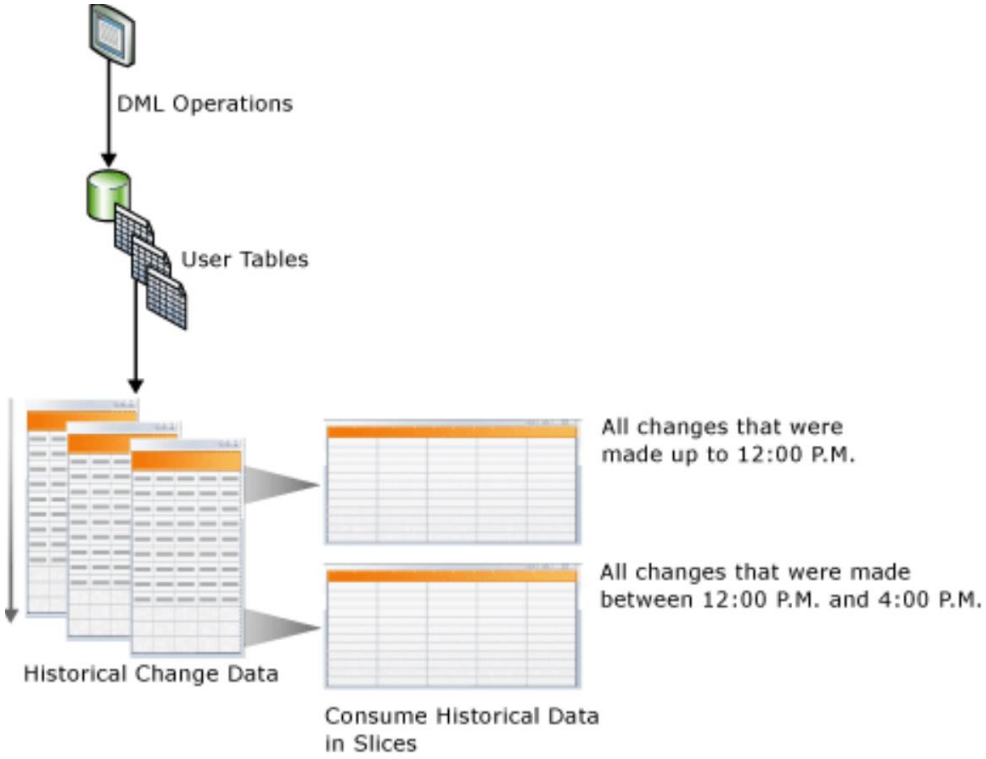

Figure3: Temporal Slicing of Historical Change Data [13]

The Lean-oriented ETL pipeline is built around three logical layers, which help to shorten the path from the raw event to the finalized metric. Such a minimalist scheme eliminates lengthy modeling stages and enables the release of an SQL prototype almost concurrently with the rollout of a new product feature.

At the Staging layer, data are copied into raw tables without any transformations; the objective is to capture the original state of events and ensure reproducibility. Modern cloud and on-premise DBMSs optimize loading operations, so fresh records appear in the database almost immediately after publication by the source, eliminating pauses between the event and its analysis.

The Transform layer is implemented via views and user-defined functions. Calculation logic remains in the query language itself, so modifying a metric reduces to recreating the view, and results are recalculated automatically. This helps avoid application rebuilds and lessens reliance on deployment, which is very critical amid frequent product experiments. It is this denormalized Data Mart layer that becomes accountable for fast analyst queries. It only stores aggregations and most commonly used slices, hence remaining compact, and can be rebuilt multiple times a day. Data retention controls help to avoid redundant volumes from building up and keep read performance stable.

Orchestration of the Lean pipeline begins with simple schedules, but as load grows, it typically transitions to specialized schedulers such as Airflow or Prefect. These provide transparent descriptions of job dependencies, allow resource-level concurrency limits, and simplify horizontal scaling. As a result, the complete cycle from event arrival to metric update fits within a short interval, sufficient for the team to make decisions based on fresh data.

In a simplified prototype, all processing is constructed on three layers, each described by direct SQL code. At the input Staging layer, data are copied into a fact table unchanged, which captures the original state of events and allows recalculation if needed. Next, the Transform layer implements business rules through a view: it cleanses data, converts measures to the required units, and computes intermediate fields for analysis. Finally, the denormalized Mart layer aggregates data to the level needed for the product solution; here, daily GMV is calculated, ready for dashboard ingestion. An example of the code is shown in Fig. 4.

```sql
-- Staging
CREATE TABLE staging_orders AS
SELECT
    event ->> 'order_id'  AS order_id,
    event ->> 'user_id'   AS user_id,
    (event ->> 'created_at')::timestamptz AS created_at,
    (event ->> 'amount')::numeric         AS amount_raw
FROM kafka.orders_topic
WHERE _partition_time >= now() - interval 'one day';


-- Transform
CREATE OR REPLACE VIEW v_orders_clean AS
SELECT
    order_id,
    user_id,
    created_at,
    amount_raw / cast('one hundred' AS numeric) AS amount,
    date_trunc('day', created_at) AS order_day
FROM staging_orders
WHERE amount_raw > cast('zero' AS numeric);

-- Mart
CREATE TABLE mart_gmv_daily AS
SELECT
    order_day,
    sum(amount) AS gmv
FROM v_orders_clean
GROUP BY order_day;
```

Figure 4: Modular ETL Pipeline for Real-Time Order Data: Staging, Cleansing, and Daily Gross Merchandise Value Aggregation (compiled by author)

Loading, cleansing, and aggregation operations are executed on schedule, with the whole cycle from event arrival to metric table update completed within fifteen minutes. Such an interval allows the team to promptly observe the effect of a marketing campaign: changes in GMV are recorded almost in real time, and comparison with the baseline indicates whether the campaign boosted sales. Engineers do not need to rebuild services; it suffices to modify the Transform view to add new segmentation logic, and within minutes, the updated figures appear in the report, supporting the rapid Lean-style hypothesis-testing rhythm.

The ephemeral data store becomes a logical extension of the minimal-product concept. While the team tests a hypothesis, it is essential to obtain data quickly and cheaply, rather than to maintain heavy infrastructure designed for a mature business. Engines that can be launched in minutes and stopped just as quickly are suitable for this task. DuckDB is convenient when the event volume fits on a developer's local disk or in the memory of a small virtual machine, providing interactive queries and no network latency. ClickHouse is chosen when more intensive reads must be served, for example, dashboards updated before each team meeting; its columnar format and compact row storage save both time and budget. BigQuery Sandbox is applied when a serverless model is desired: tables reside in the cloud and fees are charged only for actual operations, minimizing the risk of unexpected overage. At the same time, sandbox limits discipline the team by filtering out overly heavy queries.

The short lifespan of experiments necessitates strict data-volume management. A retention policy is implemented at the table level: the source writes events with an age tag or date partition, and the warehouse automatically deletes records once they exceed the configured boundary. This mechanism achieves two objectives at once: it frees space without engineer intervention and ensures that analysts work only with current information. Suppose some data must be retained for a regulatory audit. In that case, it is transferred to cold object-storage, preserving the ability to restore it without overloading the production cluster or hindering fast experiments.

When the hypothesis is confirmed and the metric becomes part of operational reporting, a need arises to migrate computations from the temporary solution to a stable platform. The migration process follows the principle of infrastructure as code: all tables, views, and user-function definitions already reside in version control, so migration consists simply of pointing them at the new environment. Initially, a dual-write setup is employed, so that data flows simultaneously into both the temporary and the target warehouse; results of corresponding queries are then compared to ensure no discrepancies or that any differences remain within acceptable bounds. Once validation succeeds, the temporary cluster is decommissioned and its resources returned to the pool, reducing costs. This approach preserves metric continuity: dashboard users do not notice the migration, since the visualization layer continues to reference the same views, now hosted on the new platform.

Notably, the transformation logic itself remains unchanged; there is no need to rewrite it from a programming language into a procedural integration language, as it is already expressed in standard SQL. Only the inclusion of production-grade quality mechanisms is required: schema validation, latency monitoring, and alerts for empty result sets. Consequently, the team retains the speed characteristic of the Lean phase, while adding the resilience and control expected in a mature analytics ecosystem. This smooth transition avoids a large big-bang deployment: resources are invested only after the metric's value has been proven, and the risk of expending funds on an unused report is effectively eliminated.

Thus, leveraging SQL prototypes to construct a rapid ETL pipeline and deploy a temporary data warehouse enables the team to minimize time and resource expenditures during early product-hypothesis validation, while maintaining sufficient data reliability and transparency; such a lightweight and flexible architecture ensures prompt measurement of key metrics and a seamless migration to large-scale solutions once value is confirmed, thereby laying a solid foundation for conclusions and further recommendations.

The experimental results reported in this study substantiate the central claim that lightweight SQL prototypes can materially accelerate the measurement-and-learning loop in early-stage product development. Empirical instrumentation across DuckDB, ClickHouse and BigQuery Sandbox demonstrates that a tightly scoped three-layer pipeline (Staging, Transform, Data Mart), implemented predominantly as declarative SQL views and scheduled jobs, is capable of producing actionable, dashboard-ready metrics on a timescale that approaches the operational thresholds required for iterative experimentation (the author reports an end-to-end update interval on the order of minutes rather than days). This outcome reinforces the argument that tactical, temporary data stores can preserve the interpretability and reproducibility of metric definitions while delivering the latency characteristics necessary for rapid, validated learning.

At the same time, the results warrant cautious interpretation because throughput, cost, and latency are highly dependent on context-specific factors that were not exhaustively varied in the presented experiments. The instrumental performance analysis provides an instructive proof of concept. Still, its external validity will depend on workload characteristics (event cardinality, join cardinality, retention windows), concurrency patterns, and the operational constraints of particular cloud providers or on-premise environments. Moreover, regulatory, security, and governance requirements may necessitate adaptations to the ephemeral architecture (for example, implementing hardened access controls or establishing permanent archival paths), which could introduce additional latency or operational overhead not captured in the current measurements.

Taken together, the findings suggest a pragmatic pathway for practitioners and researchers alike: adopt SQL-centric prototypes for hypothesis validation to minimize calendar and budgetary friction, while rigorously measuring and documenting the transform logic, latency, and cost trade-offs in situ. Future work should focus on systematic benchmarks across a broader set of workloads, formal cost-benefit models comparing ephemeral versus traditional DWH approaches, and longitudinal studies that quantify the operational debt incurred (or avoided) by repeated short-lived experiments. Such follow-up would both refine the boundary conditions under which the Lean SQL prototype approach is preferable and produce prescriptive guidance for migrating validated metrics into robust production analytics systems.

4.Conclusion

This has validated an Agile method of early validation of product hypotheses through the use of lightweight SQL prototypes, temporary data stores, and simplified ETL pipelines. It proves that leveraging existing DBMSs together with the declarative SQL language empowers teams to implement minimally viable data pipelines without long procurement or approval processes — thereby eliminating the calendar barrier typically baked into traditional DWH initiatives. A short cycle from event arrival to finalized metric minimizes the risk of analytical data going stale and accelerates decision-making on fresh facts.

Economic analysis confirmed that the minimalist architecture reduces budgetary risks typical of large IT projects: the use of temporary clusters and automatic data-retention policies provides cost control, and serverless sandbox solutions limit unintended overspending. Moreover, the widespread availability of SQL and transparency of query code engage a broad spectrum of product stakeholders—from developers to managers and marketers—improving

collaboration and enhancing the quality of validated learning.

The technical implementation of the Lean pipeline via Staging, Transform, and Data Mart layers has demonstrated that a modular ETL pipeline enables modification of calculation logic with a single 'create or replace view' statement, eliminating the need to rebuild or redeploy services. As a result, the team can adapt to new experiments within minutes, preserving both the reproducibility of source data and the flexibility to migrate computations to a mature platform under the infrastructure-as-code paradigm. A clean transition from temporary to permanent data stores helps keep metrics available and reduces operational risks as scaling takes place.

To provide a balanced perspective, the scope of this study and its limitations should be clearly defined. The experimental scope was intentionally focused on validating the core methodology across a representative set of prototype engines (DuckDB, ClickHouse, BigQuery Sandbox) using specific ETL patterns. While the results establish a strong performance baseline, future research could expand upon these findings by validating the approach across a broader spectrum of workloads, including those with higher event cardinality, complex joins, and high-concurrency access patterns. Similarly, the study isolates the core pipeline logic from operational variables such as network topology or specific storage configurations. A subsequent investigation could analyze the sensitivity of these results to such infrastructure-specific factors in production environments.

From a methodological standpoint, the study's focus is on the initial validation phase of a product hypothesis. Consequently, important topics such as the long-term accumulation of technical debt from repeated experiments or the integration with specific compliance frameworks (e.g., GDPR, PCI) in regulated domains are identified as valuable but separate areas for future longitudinal studies. The cost analysis is centered on the direct economics of temporary infrastructure; a comprehensive total cost of ownership (TCO) model, including indirect costs like engineering time for migration and monitoring, would be a logical extension to this work. Finally, this paper adopts a techno-economic perspective. The organizational dynamics that influence the adoption of such prototypes—including team skills, change management, and cross-functional coordination—represent a rich field for complementary research.

To sum up, using SQL prototypes, temp DWHs, and fast ETL chains provides a solid foundation for quick product-hypothesis checking by combining the speed of Lean with the needed levels of data quality and control. Such a plan can be used as a tool for startups and early-stage product teams to align the strength of infrastructure resource use with the need for timely, accurate metrics.

References

[1] J. York, N. Turner, and S. Hussels, “Lean Startup and Learning Loops in Entrepreneurial Ventures: A Systematic Review,” *Journal of Knowledge Management and Practice*, vol. 24, no. 1, 2024, Accessed: Jun. 29, 2025. [Online]. Available: https://journals.klalliance.org/index.php/JKMP/article/download/201/196

[2] S. Bryant, “How many startups fail and why?” *Investopedia*, Jun. 24, 2024. https://www.investopedia.com/articles/personal-finance/040915/how-many-startups-fail-and-why.asp

(accessed Jun. 30, 2025).

[3] D. Quintero, L. Bolinches, A. Gandakusuma, S. Nicolas, J. Reinaldo, and T. Katahira, "Redbooks IBM Data Engine for Hadoop and Spark," IBM, 2016. Accessed: Jul. 01, 2025. [Online]. Available: https://www.redbooks.ibm.com/redbooks/pdfs/sg248359.pdf

[4] P. Gray and C. Israel, "The Data Warehouse," *University of California*. https://escholarship.org/content/qt1hp1k5m7/qt1hp1k5m7.pdf (accessed Jul. 01, 2025).

[5] I. Kravchenko, "Data Warehouse Implementation: 10 Tips to implement DWH for a Bank," *DICEUS*, Sep. 27, 2021. https://diceus.com/implement-data-warehouse-bank-9-months/ (accessed Jul. 02, 2025).

[6] Airbyte, "How to Build a Data Warehouse from Scratch," *Airbyte*, 2025. https://airbyte.com/data-engineering-resources/building-data-warehouse (accessed Jul. 03, 2025).

[7] Datavault Builder, "Enhancing Retail Operations with Data Vault-based Data Warehouse Automation," *Datavault Builder*, Nov. 29, 2024. https://datavault-builder.com/use-cases-retail/ (accessed Jul. 03, 2025).

[8] M. Bloch, S. Blumberg, and J. Laartz, "Delivering large-scale IT projects on time, on budget, and on value," *McKinsey*, Oct. 01, 2017. https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/delivering-large-scale-it-projects-on-time-on-budget-and-on-value

[9] "Software Projects Don't Have to Be Late, Costly, and Irrelevant," *BCG Global*, Apr. 25, 2024. https://www.bcg.com/publications/2024/software-projects-dont-have-to-be-late-costly-and-irrelevant (accessed Jul. 04, 2025).

[10] M. Ashare, "Cloud data storage woes drive cost overruns, business delays," *CIO Dive*, Feb. 26, 2025. https://www.ciodive.com/news/cloud-storage-overspend-wasabi/740940/

[11] "Technology - 2024 Stack Overflow Developer Survey," *Stack Overflow*, 2024. https://survey.stackoverflow.co/2024/technology#most-popular-technologies-language-prof (accessed Jul. 04, 2025).

[12] I. Belcic and C. Stryker, "Self-Service Analytics," *IBM*, Sep. 04, 2024. https://www.ibm.com/think/topics/self-service-analytics (accessed Jul. 05, 2025).

[13] M. Ray, "Track Data Changes - SQL Server," *Microsoft*, Sep. 19, 2024. https://learn.microsoft.com/en-us/sql/relational-databases/track-changes/track-data-changes-sql-server?view=sql-server-ver17 (accessed Jul. 06, 2025).