Materialize

WHITEPAPER

Materialize vs ClickHouse: Comprehensive Enterprise Platform Analysis

Companies today need real-time operational intelligence for fast-changing data. They also need high-performance analytics for their ever-increasing volumes of historical data. Legacy “one-size-fits-all” database approaches can’t keep up, and business intelligence tools that attempt to fill these needs are often both slow and expensive. Instead, enterprise organizations are turning to thoughtfully composed architectures that leverage specialized systems to address their increasingly complex — and increasingly mission-critical — data needs.

These organizations recognize that real-time operational analytics and historical analytical processing are fundamentally different problems requiring different solutions. The way forward for many use cases may be to harness complementary technologies that together can meet the full spectrum of modern data platform requirements (while also delivering superior ROI over other systems and solutions).

Understanding the core technologies

Materialize and ClickHouse are both high-performance analytical data systems, but they are designed for fundamentally different workloads and architectural patterns. We will examine the core differences in architecture, materialized view support, consistency, data mutability, and intended use cases for both. We will also look at some examples of how they can be used in conjunction to deliver highly performant, always up to date, and strongly consistent data analytics for a variety of organizational data needs.

Materialize is a live data layer for apps and AI agents that incrementally maintains the results of SQL queries over continuously changing upstream data. It is optimized for live, stateful workloads, offering strong consistency guarantees and full support for inserts, updates, and deletes, and helping AI agents stay in sync with reality.

ClickHouse is a high-performance columnar OLAP database designed for executing analytical queries over large volumes of immutable data. It is optimized for fast, ad hoc queries on append-only datasets for exploring and analyzing existing historical data.

ClickHouse: High-performance analytics engine

ClickHouse is a data engine purpose-built for answering complex questions about large datasets, very quickly. Clickhouse does multiple things well:

- **Blazing fast analytics:** Columnar storage, aggressive compression, and vectorized execution enable sub-second queries across billions of rows of data.
- **Scalable data storage:** Efficiently handles petabytes of historical data with predictable performance characteristics.
- **Ad hoc query excellence:** Optimized for exploratory analytics where query patterns are unpredictable.
- **Cost-effective storage:** Excellent compression ratios reduce storage costs for long-term data retention.
- **Resource efficiency:** ClickHouse's performance can lead to savings on compute resources.

ClickHouse is optimized for classic analytical workloads like historical data trend analysis and business intelligence reporting. But it's also a strong observability asset for queries and powerful aggregations on logs, metrics, traces, session replays and errors catching, even on high-cardinality data. Companies have used ClickHouse for financial reporting and compliance analytics and to examine customer behavior across giant datasets.

Materialize: The live data layer for apps and AI agents

Materialize is a streaming database that maintains incrementally updated SQL query results over continuously changing data. Unlike traditional databases that compute results on demand, Materialize continuously maintains the answers to your queries. Materialize does many important functions exceptionally well:

- **Incremental view maintenance:** Updates query results as upstream data changes, keeping data views current without expensive batch reprocessing.
- **Strong consistency:** Guarantees strict-serializable consistency across all data sources and queries.
- **Complex real-time views:** Supports multi-way joins, aggregations, window functions, and even recursive SQL on live changing data.
- **Immediate reactivity:** Single digit millisecond latency for querying pre-computed results.
- **Flexible data ingestion:** Can ingest data from multiple sources in varying formats, acting as a transformation layer to transform, aggregate, and normalize disparate data and then creating a materialized view, continually updated in real-time as upstream data changes.

Materialize's primary purpose is real-time data integration and transformation. This makes it ideal for applications like fraud detection and alerting systems; supply chain visibility and exception handling; personalized, customer-facing analytics; and even regulatory compliance monitoring with immediate alerting.

It also makes Materialize the perfect partner for agentic AI applications. AI agents need real-time, accurate models of business data with fraction-of-a-second updates whenever data changes so they always reflect reality when the LLM queries them. Materialize's Incremental View Maintenance combines with Materialize's flexible data ingestion capabilities to transform less-refined data into an always-current model – at scale – that AI agents can easily discover and use.

THE UPSHOT:

ClickHouse is optimized for analytical workloads where you want to store and query very large data sets that do not receive constant updating. Materialize, on the other hand, is optimized for workloads that require frequent updates and recomputation. Materialize is where you want to keep your fast-changing data (or any other data that powers any kind of real-time or operational workload) and context-retrieval workloads for agentic AI.

Materialize vs. ClickHouse: Comparative analysis

Both systems support SQL interfaces, integrate with modern data stacks (dbt, BI tools), and can handle complex analytical workloads. Other capabilities, however, diverge due to Materialize and ClickHouse having respective designs and primary purposes.

Materialized views

Unlike regular views, which are virtual and dynamically generated each time they're accessed, materialized views precompute a query and then physically store the results for lowest possible latency. This precomputation is especially effective when you have complex queries or deal with massive datasets, because materialized views remove the need for repeated computations.

Clickhouse and Materialize both support materialized views, but the design and capabilities differ substantially.

Materialize

Materialize allows you to create materialized views using standard PostgreSQL SQL syntax. You can write complex SQL queries — with joins, aggregations, subqueries, window functions, etc. — and Materialize will continuously maintain the results of those queries even as upstream data changes.

Unlike many other systems that restrict the types of queries you can materialize, Materialize supports very sophisticated SQL operations in materialized views, such as:

- Multi-table joins across different data sources, including between streams and reference tables
- Nested aggregations and grouping
- Window functions and subqueries for ranking and time-series analysis
- Complex WHERE clauses and filters
- Common table expressions (CTEs)
- Recursive CTEs for graph traversals and hierarchical aggregations

This is important because traditional materialized views in most databases are quite limited — they might only support simple aggregations or have restrictions on joins. Materialize, on the other hand, lets you take any complex analytical query you'd run in PostgreSQL and turn it into a continuously updated materialized view. If you know PostgreSQL, you can immediately start creating materialized views in Materialize.

Once you define a view, Materialize keeps it incrementally maintained: Results are updated in response to new data, updates, or deletes that come in from any of your input sources. All these changes are applied consistently and in transactional order. Your users get to build reactive, always-up-to-date queries that remain correct as upstream data changes.

ClickHouse

ClickHouse is a batch platform, so it runs computation on read: when you send a query to ClickHouse to call data back, that's when the computation is done. ClickHouse is really efficient and you'll get back really fast queries, but ClickHouse can't do incremental computation. This means that, as upstream data changes, they can't keep that data fresh.

As a result, ClickHouse does support materialized views, but with key limitations:

- Materialized views are defined on a single source (the fact table), and are updated only when new data is inserted into that source.
- Views do not respond to updates or deletes in the source table.

- Joins to dimension tables are evaluated at insert time; subsequent changes to those dimension tables do not affect the materialized view.
- There is no mechanism for automatic invalidation or recomputation of views based on changes outside the base table.

Ultimately, ClickHouse materialized views are well suited for event rollups and simple aggregations triggered by inserts. These views, though, are not suitable for modeling evolving state or maintaining consistency across multiple, continually changing tables.

Consistency

Materialize

Materialize provides strong consistency guarantees, enforcing serializable consistency across all sources and queries. Materialize's strong consistency is essential because, under Materialize's performance model, new inputs are applied incrementally without full recomputation. Even though there may be many moving parts when a query comes in, Materialize ensures that the numbers are always going to be strictly correct.

- Queries reflect a consistent snapshot of all input data at a specific logical timestamp.
- Updates and deletes are handled incrementally and applied in the correct transactional order.
- Multi-stream joins produce correct results without requiring users to manage timing or coordination.

Strong, serializable consistency makes Materialize appropriate for workloads where correctness under change is required, including operational analytics, data products, and embedded real-time business logic.

In Materialize, the data engine is focused on consistency and building out composable data products where you can have views stacked on top of each other, join them together, and create your ideal custom analytics dashboard. With strong consistency out of the box, you know that if you have multiple items on your visualization, say two summaries of your data that break it out in different ways, Materialize will guarantee that when the page loads, they're both showing you results as of the exact same input datasets. Unlike other systems where different frequencies of ETLs can produce a lot of inconsistency, Materialize's data will always tie out.

It also makes Materialize an ideal data source for AI agents and MCPs — one they can access without taxing production systems, and with consistency guarantees all the way across. This emerging digital twin architecture makes it possible to have an exact, always-current model of your relevant business entities and their relationships, expressed in the language of your company (customers, orders, suppliers, routes) rather than low-level tables, that AI agents can query against. **Materialize's strong consistency helps agentic AI applications stay in sync with reality by immediately reflecting any changes that happen as a consequence of an action taken by the agent.**

ClickHouse

Because ClickHouse prioritizes throughput and analytical performance over consistency, it can only offer eventual consistency in both distributed deployments and materialized view maintenance:

- Queries may observe partially updated or inconsistent states when joining across changing tables.
- Materialized views are not updated in response to changes in reference data or dimension tables.
- Asynchronous replication means that distributed nodes may temporarily reflect different states of the data.

That said, ClickHouse is optimized for append-only data models where strong consistency across updates is not a core requirement. This is why ClickHouse is not typically recommended for transactional applications, or any other workloads that require frequent updates and row-level operations.

Data Mutability

Data mutability describes data's ability to change after it's been stored. "Mutable" data can be modified, while "immutable" data cannot be changed once written.

Materialize

One of Materialize's key strengths is how it handles changing data: Materialize supports full data mutability. It can ingest:

- **Inserts:** Adding new records (like a new customer signup)
- **Updates:** Modifying existing records (like updating a customer's address)
- **Deletes:** Removing records (like canceling an account)

These changes may come from directly running SQL commands against Materialize to modify data, just like a regular database. Materialize also offers a more powerful capability: streaming data sources with Change Data Capture (CDC). CDC lets you capture every change that happens in your upstream data sources (like PostgreSQL or MySQL) and then stream those changes to Materialize in real time.

This is what makes Materialize uniquely powerful for real-time data systems: **when data changes, all dependent views are updated immediately and correctly.**

In most systems, if you have a materialized view that aggregates data, and then upstream data changes, you'd need to manually refresh the view or wait for a scheduled refresh. Materialize, however, tracks logical records (usually through primary keys) so that it understands the "identity" of your data records.

So when, for example, it sees an UPDATE for customer ID 12345, it knows this relates to the same customer record it processed before, not a new customer. This allows Materialize to do incremental updates, modifying only the changed data itself instead of recomputing everything. Materialize's strict consistency then ensures these incremental changes are instantly, globally, and consistently applied to maintain correctness across all views. This is fundamentally different from systems where you'd need to rebuild the entire view or manually manage these cascading updates.

ClickHouse

ClickHouse is built as a data warehouse—a system designed for analyzing large amounts of data, not for day-to-day operational tasks. Its OLAP focus (Online Analytical Processing) means it's optimized for complex queries that analyze trends, patterns, and aggregations across large datasets. As a result, ClickHouse is fundamentally designed to handle append-only workloads where you primarily add new data rather than changing existing data.

ClickHouse's columnar storage architecture drives its performance optimization for immutable workloads, like long-term historical analysis and data exploration. Instead of storing data row-by-row (like traditional databases), ClickHouse stores data column-by-column. This makes it incredibly fast for analytical queries but more complex for updates.

Thus, while ClickHouse does support SQL modifications like `ALTER TABLE UPDATE` and `DELETE`, these operations are not ideal for workloads where data changes frequently

Asynchronous background merges: When data gets updated, ClickHouse doesn't process the changes immediately – it incorporates the change later, during maintenance cycles.

No materialized view updates: Data changes do not trigger updates in dependent materialized views. If you have a summary table (materialized view) showing "total sales by region" and you change the region where a given sale occurred, ClickHouse won't automatically update that summary data view. You'd need to manually refresh it.

Resource intensive: Updates require ClickHouse to rewrite entire data blocks, which can slow down the whole system and introduce delays before changes are visible.

As a result, ClickHouse excels at answering "what happened?" questions about large amounts of historical data, such as *Which customers generated the most revenue in 2023?* or *How did our website traffic change during the holiday season?*

ClickHouse struggles with "what's happening now?" scenarios that require frequent data updates, like real-time inventory tracking where stock levels constantly change, or dashboards tracking current status for manufacturing operations.

Performance model

ClickHouse and Materialize are two systems that achieve performance in fundamentally different ways, and for fundamentally different analytic purposes.

Materialize's performance model: "Always ready"

Materialize is optimized for incremental computation: Instead of recalculating everything from scratch when data changes, Materialize only updates the parts that are affected. More importantly, once you define a materialized view, its results are maintained continuously.

New inputs are applied incrementally without full recomputation: Again, instead of recalculating everything from scratch when data changes, Materialize updates only the affected data.

Near-zero query latency for materialized views: Because the results are already computed and stored, querying a materialized view is just reading pre-calculated data. A complex query that might take 30 seconds in a traditional database returns instantly in Materialize because the answer is already ready.

Performance scales with volume of changes, not data size: Whether you have a thousand records or one billion, if only 100 records change, Materialize only processes those 100 changes. In traditional data analytics systems, performance degrades as data grows because they must scan all data every time. Materialize's performance depends on how much your data is changing, not your total data volume.

Dynamic data platform for AI agents: Materialize's incremental computation model also enables "agent-ready" data infrastructure so you can handle the massive increase in queries when moving from human to machine traffic.

Materialize's performance model is well-suited for operational systems where specific, predefined queries must remain current with the lowest possible latency. Example use cases include live fraud detection where flagged transactions need immediate alerting, or live shipment tracking for a logistics company.

It's also the first system that can give AI agents the real-time, semantically rich business context they need to operate effectively while simultaneously supporting massive, agent-scale workloads.

ClickHouse's performance model: "Complex queries, fast"

ClickHouse is optimized for high-throughput, on-demand analytical queries. This means ClickHouse can answer complex analytics questions over immense quantities of data very quickly when you ask them, but it doesn't maintain these answers between queries.

Fast scan performance over large datasets: Data is organized by column, so if you want to sum all sales amounts, ClickHouse only reads the "`sales_amount`" column, not entire rows. ClickHouse also uses CPU optimizations to process many values at once; and leverages efficient data compression to reduce I/O because similar values in columns compress very well.

Full or partial table scans: ClickHouse can read through large portions of your data to answer queries with very low latency. To answer "average order value by month," for example, ClickHouse might scan millions of order records but do it so efficiently that it still completes in seconds.

Ad hoc queries: You can ask ClickHouse questions it's never seen before, and it will figure out how to answer them efficiently. This contrasts with Materialize, which needs you to define your queries upfront as materialized views.

No saved state between queries: ClickHouse doesn't "remember" the results of previous queries. Each query starts fresh. This is great for exploratory analysis where you're asking different questions each time, like BI workloads over static or slowly changing data, but is not suitable for real-time workloads like dashboards or alerting.

Why consistency, data mutability, and performance matter for architecture decisions

Understanding the comparative strengths and limitations for Materialize and ClickHouse as high-performance analytical systems helps explain why many enterprises need both systems. **The key insight is that these aren't competing approaches — they're complementary strategies for different types of data workloads.**

ClickHouse excels at historical analysis and reporting where:

- You're doing exploratory data analysis
- Query patterns are unpredictable/ad hoc
- You're working with large historical datasets
- You can tolerate some delay while queries execute

Materialize excels at live operational analytics where data changes frequently and:

- You have a specific set of metrics that need to always be current
- Users expect instant response times
- You need data changes reflected immediately
- You're supporting agentic AI applications

Many enterprise organizations, though, need both real-time operational metrics (Materialize) AND performant historical analysis capabilities (ClickHouse).

The key is matching the right tool to the right job, rather than trying to force one system to do everything.

Materialize & ClickHouse: Better Together

Two real-world case studies of organizations using ClickHouse and Materialize in conjunction: the analytics challenges they faced, and how they solved these by adopting both systems to work together.

Case Study: A SaaS supply chain operating system for logistics and transportation

THE COMPANY:

A startup that built a supply chain operating system, offered as a SaaS platform. Despite being a small company, they serve major enterprise clients and are experiencing rapid growth in the logistics and transportation industry.

THE CHALLENGE:

Complex multi-client operations: The platform serves logistics businesses that have complex supply chain operations. For example, one customer is a trucking company that manages deliveries for 20+ different clients on a single truck, each with unique routes, loads, and delivery requirements.

Dual analytics requirements: The company needed both route optimization capabilities (analyzing largely static data to plan efficient delivery routes) and real-time tracking services (immediate alerts when conditions change).

ClickHouse limitations: The platform was launched using ClickHouse for analytics across dozens of millions of data points generated by their users. ClickHouse was great for logistics, generating optimized delivery routes for each truck's unique load which, of course, changed day to day. But the startup struggled with providing a real-time tracking service on their platform — a feature that was in high demand from customers — because real-time incrementally updated views are just not an out-of-the-box functionality ClickHouse offers.

Custom workaround problems: The startup's developers built custom, fragile workarounds to update data and trigger notifications (a common pattern with organizations trying to force ClickHouse into real-time use cases) but they knew they needed a purpose-built solution that could reliably trigger downstream notifications whenever upstream data changed — especially given tight customer SLAs.

Rejected alternatives: Full-featured BI tools were evaluated but rejected as too slow and not customizable enough for their embedded platform needs.

The solution architecture:

Phase 1: Route optimization (ClickHouse)

- **SLA Compliance:** Ensures tight delivery commitments are monitored and communicated in real-time
- **Delivery Planning:** Analyzes historical and current data to generate optimized routes for each truck's unique daily load
- **Performance Analytics:** Supports exploratory analysis and ad hoc reporting on delivery patterns
- **Historical Insights:** Enables long-term trend analysis for route efficiency improvements

Phase 2: Real-time alerting system (Materialize)

- **Live shipment tracking:** Continuously ingests and processes logistics information as deliveries progress
- **Threshold monitoring:** Automatically detects when shipments exceed time/location parameters
- **Immediate Notifications:** Triggers real-time alerts to affected clients when exceptions occur (breakdowns, delays, early arrivals)

Phase 3: Integrated logistics platform

- **Seamless integration:** Both systems are Postgres wire compatible, enabling quick integration into existing analytics workflows
- **Eliminated custom code:** Replaced fragile workaround solutions with consistent, real-time views capabilities
- **Embedded analytics:** Delivered real-time alerting as a native platform feature rather than a bolt-on solution

Why this required both systems:

ClickHouse alone: Perfect for route optimization and historical analytics but fundamentally unable to provide the real-time incremental updates required for tracking services, leading to time-consuming custom development.

Materialize alone: Could handle real-time alerting excellently but wasn't optimized for the large-scale route optimization analytics that ClickHouse provided.

Together: Created a complete logistics intelligence platform using ClickHouse to analyze delivery route optimizations based on largely immutable data, and Materialize to ingest a shipment's logistics information and continually update it, make sense of it, and to trigger an alert or other actions as soon as a threshold is met or not met.

This case study shows how ClickHouse and Materialize are not duplicate tools and why they work so well in conjunction with each other. ClickHouse acts as a great, super-performant warehouse for time series data and other things like exploratory analytics and ad hoc historical reporting. Materialize is handling the real time incremental view that continually ticks away as the transformation and real-time serving layer. Together, they keep the logistics running smoothly.

Case Study: Private equity firm risk management platform

THE COMPANY:

A private equity firm with an insurance component that needs to continuously analyze and evaluate the risk of their investment positions using data from 8-10 different insurance providers.

THE CHALLENGE:

Data integration complexity: Each insurance provider sends data in different formats, schemas, and frequencies, making it difficult to create a unified view of risk.

Real-time requirements: Risk conditions change constantly, requiring immediate recalculation of risk scores as new data arrives (ranging from every 20 seconds to hourly).

Historical analysis needs: The firm also needs to perform exploratory analysis and ad hoc queries on historical insurance data for trend analysis and regulatory reporting.

The solution architecture:

Phase 1: Real-Time Data Transformation with Materialize

- **Data ingestion:** Materialize receives insurance data from multiple providers in various formats and frequencies
- **Real-time transformation:** Automatically normalizes, cleanses, and standardizes incoming data from all sources
- **Live risk calculation:** Continuously maintains materialized views that calculate and update risk scores as conditions fluctuate
- **Immediate availability:** Risk analysts have access to current risk assessments with zero query latency

Phase 2: Historical Data Storage (ClickHouse)

- **Data aging strategy:** After 1-2 days, transformed data is bulk exported from Materialize to ClickHouse
- **Long-term storage:** ClickHouse efficiently stores historical insurance data from all providers
- **Exploratory analytics:** Enables ad hoc queries like *Show me all data from Provider X over the past 60 days*

Phase 3: Integrated logistics platform

- **Live vs. historical comparisons:** Ability to query up-to-the-minute data from Materialize against historical context from ClickHouse
- **Dynamic dashboards:** Shows current risk levels alongside historical trends (e.g., “today’s risk vs. 30-day rolling average”)
- **Unified querying:** Single interface that pulls fresh data from Materialize and historical data from ClickHouse

Why this is required of both systems:

Materialize alone: Could handle real-time transformation and risk calculation but would be prohibitively expensive for long-term historical storage and not optimized for exploratory historical analysis.

ClickHouse alone: Could hold and analyze historical data efficiently but couldn’t provide the real-time transformation, normalization, and continuous risk calculation required for operational use.

Together: Created a complete solution where each system handles what it does best, resulting in immediate operational analytics paired with comprehensive analytical depth.

The complementary strengths of Materialize and ClickHouse can solve complex enterprise data challenges that neither system can ideally address on its own. By doing this, this investment firm gained:

- **Engineering efficiency:** Each system handles what it does best, reducing workarounds and custom solutions
- **Cost optimization:** Right-sized infrastructure for different data lifecycle stages
- **Operational excellence:** Strong consistency for operational data, high performance
- **Future-proofing:** Modular achitecture that can evolve with business need

Conclusion

Materialize and ClickHouse are optimized for different workloads and serve distinct purposes in a data architecture. In many modern architectures, though, these systems can be complementary. As a live data layer for applications and AI agents, Materialize can provide accurate, real-time views that always maintain data state, while ClickHouse serves as a performant backend for historical exploration and long-term analytics.

Use Materialize when you need always-up-to-date views over streaming or mutable data, with strong consistency, low latency, and complex query support – or support for real-time context-retrieval workloads for agentic AI applications architectures.

Materialize is best suited for known, repeated queries that must always be current. It's built for live incremental views where data changes frequently and you need those changes reflected immediately, and also where users expect instant response times. Ideal Materialize workloads are uses like fraud detection, where flagged transactions need immediate alerting, or live shipment tracking for a logistics company.

AI agents need real-time, accurate models of business data with fraction-of-a-second updates whenever data changes so they always reflect reality when the LLM queries them. Materialize's incremental views and flexible data ingestion capabilities transform less-refined data into an always-current model – at scale – that AI agents can easily discover and use.

Use ClickHouse when you need dynamic, high-speed analytical queries over large volumes of immutable data, and consistency or reactivity is not a primary concern.

ClickHouse is best suited for working with datasets where records are rarely modified after insertion. ClickHouse is great for flat wide tables or historical data running ad hoc analytics. Queries like *Show me a sales report for the last month* or *Show me how many of our shipments were over 20 minutes late in the last 90 days* are ideal ClickHouse analytical workloads.

Use Materialize + ClickHouse together when you need both a strongly consistent real-time operational data store and performant ad hoc historical analysis.

Using Materialize with ClickHouse would be super efficient for live comparisons of current vs. long term data – for example, *How am I tracking today against my daily average from the last 30 days rolling?* You can use them together to create a visualization that's continually up to date with current data versus your last 30 days rolling, You've got this ultra-performant access to data from three seconds ago paired with this really efficient way to pull data from the last month or two months or three years and surfacing that in conjunction.

By embracing this complementary systems approach, enterprises can build more efficient, maintainable, and cost-effective data platforms that truly serve their business needs. ClickHouse and Materialize are a powerful combination that together can satisfy the full spectrum of your analytical data requirements — and position you for a flexible future of composable data products.

Appendix 1: Use case fit

Requirement	Materialize	ClickHouse	Materialize + ClickHouse
Streaming data support	✓ (native)	Limited (append-only)	✓
Real-time business data model support for agentic AI applications	✓	Not supported	✓
Arbitrary materialized views	✓	Limited to single-table inserts	✓
Reactivity to updates/deletes	✓	X	✓
Joins across mutable tables	✓	Not supported in materialized views	✓
Serializable consistency	✓	X	✓
Ad hoc query Performance	Adequate for simple queries	Optimized	✓
Long-term historical analysis	Not optimized	✓	✓
Real-time operational analytics	✓	Limited	✓
Immutable event analytics	Limited	✓	✓
Data mesh / live APIs	✓	X	✓

Appendix 2: Licensing and ecosystem

- **Materialize** is source-available under the BSL license. A free Community Edition is available with resource limits. For horizontal scaling, high availability, and cloud deployment, a commercial license or the hosted Materialize Cloud offering is required.
- **ClickHouse** is fully open source under the Apache 2.0 license. It can be self-hosted or used via ClickHouse Cloud or other managed offerings.

Both systems offer mature SQL interfaces, integrations with popular tools (e.g., dbt, BI tools), and active, engaged user communities.