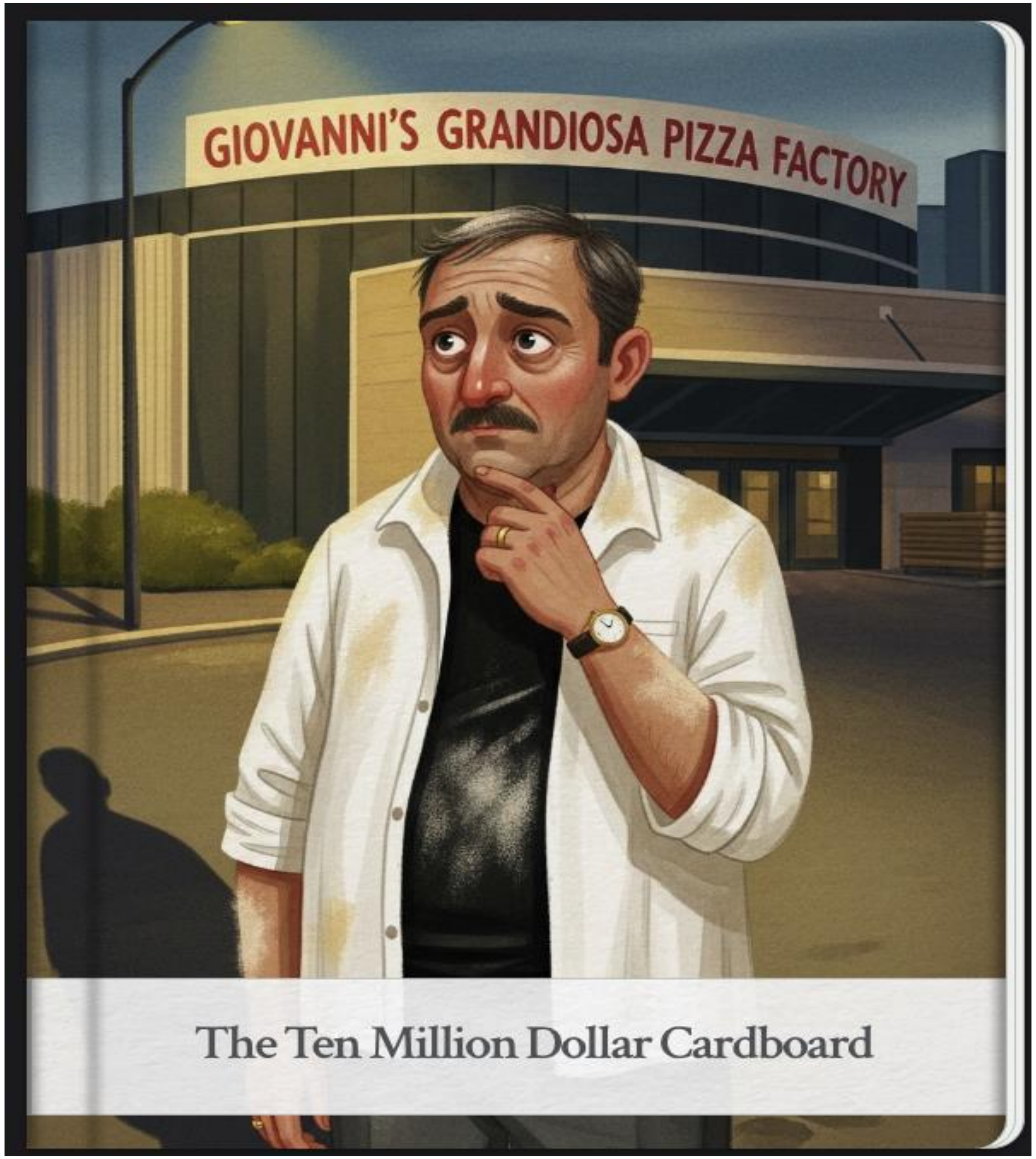
Why We Chose ClickHouse at ThriveStack

Ankit Gupta
Founding Member
ThriveStack
ankitg@thrivestack.in

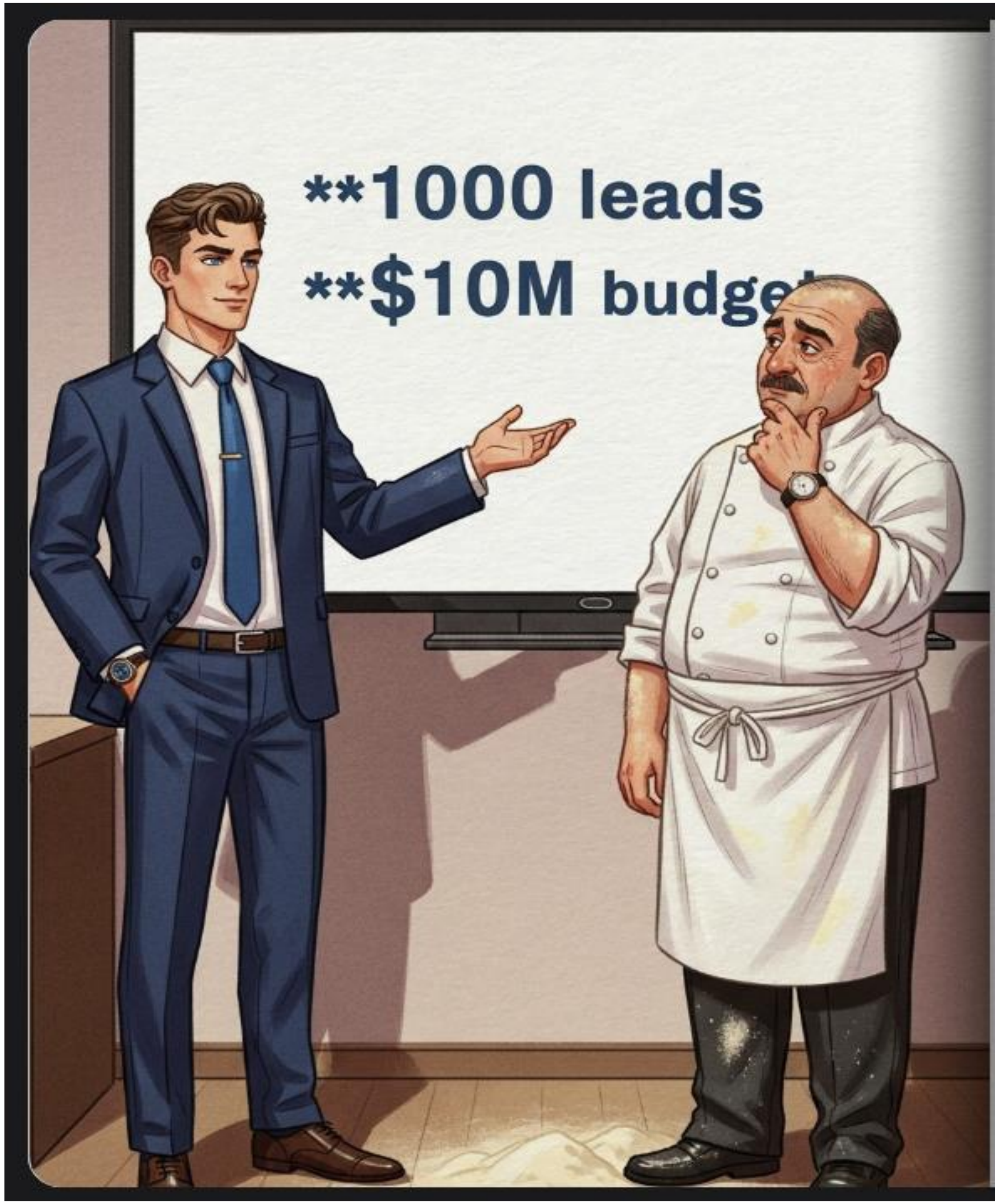
https://www.thrivestack.ai/

Agenda

1. What is ThriveStack?
2. How we chose Clickhouse ?
3. Using ClickHouse Effectively
4. Our Next Steps

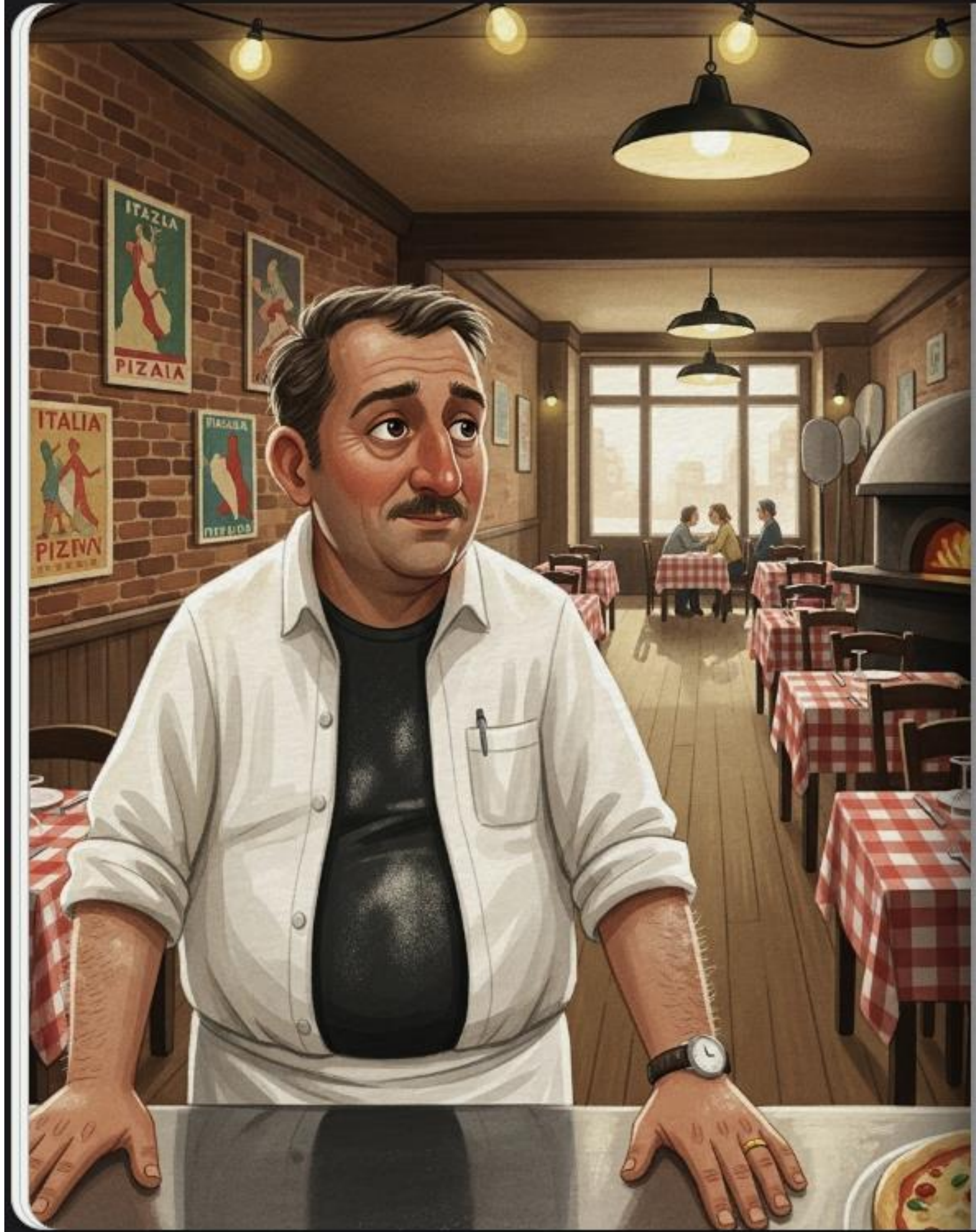


The Ten Million Dollar Cardboard

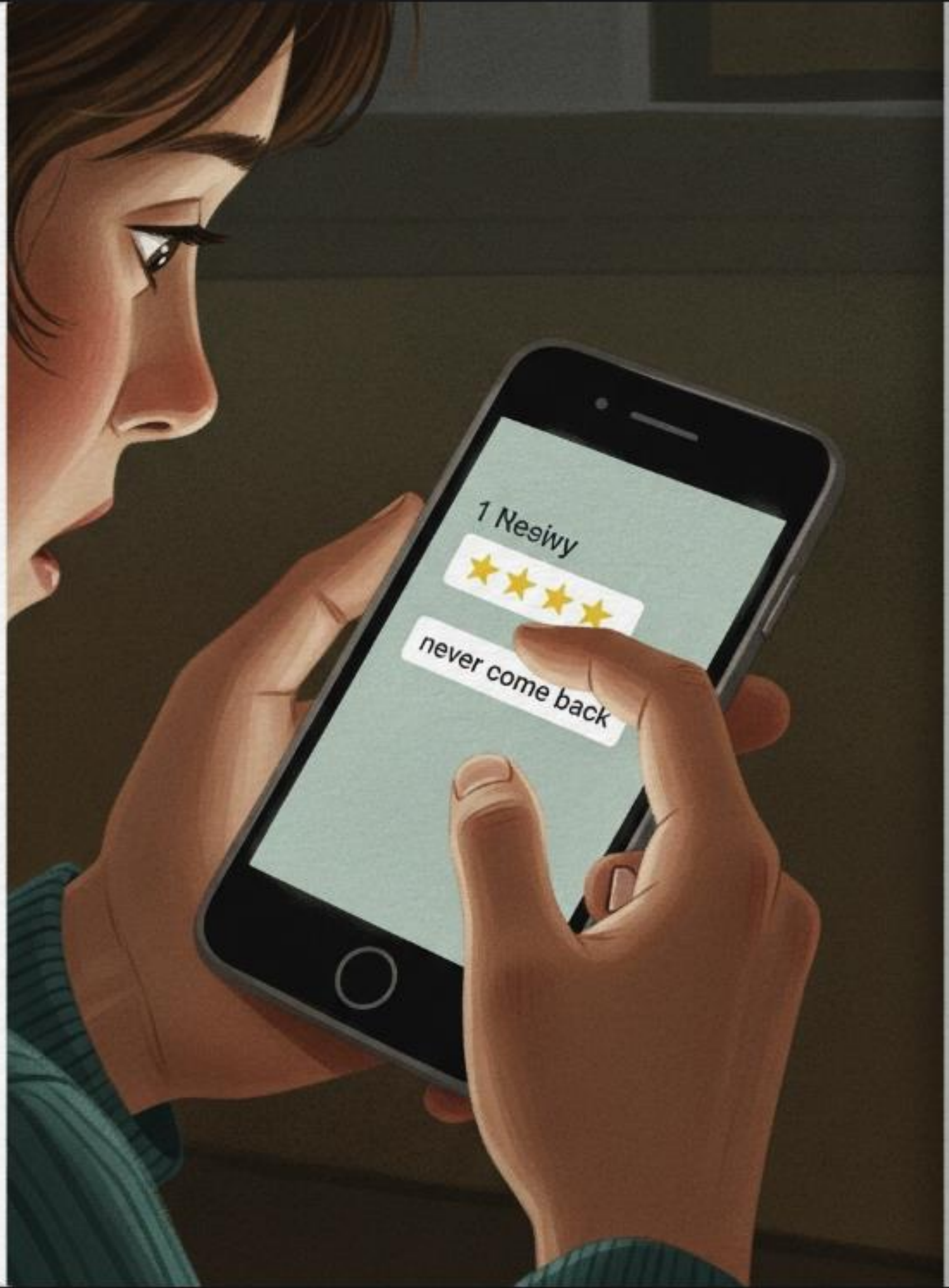


"Marketing promised you 1000 leads - you spent your $10M budget."

"But you got 500 visitors walking in."



"Only 10 of them bought pizza."



"2 of them disliked it and vowed to never come back again."

Department Victory Laps

Marketing Team

500 website visits!

"Our campaigns are smashing it - look at this traffic surge!"

Sales Team

12 orders taken!

"CRM is performing brilliantly - pipeline's never been stronger!"

Operations Team

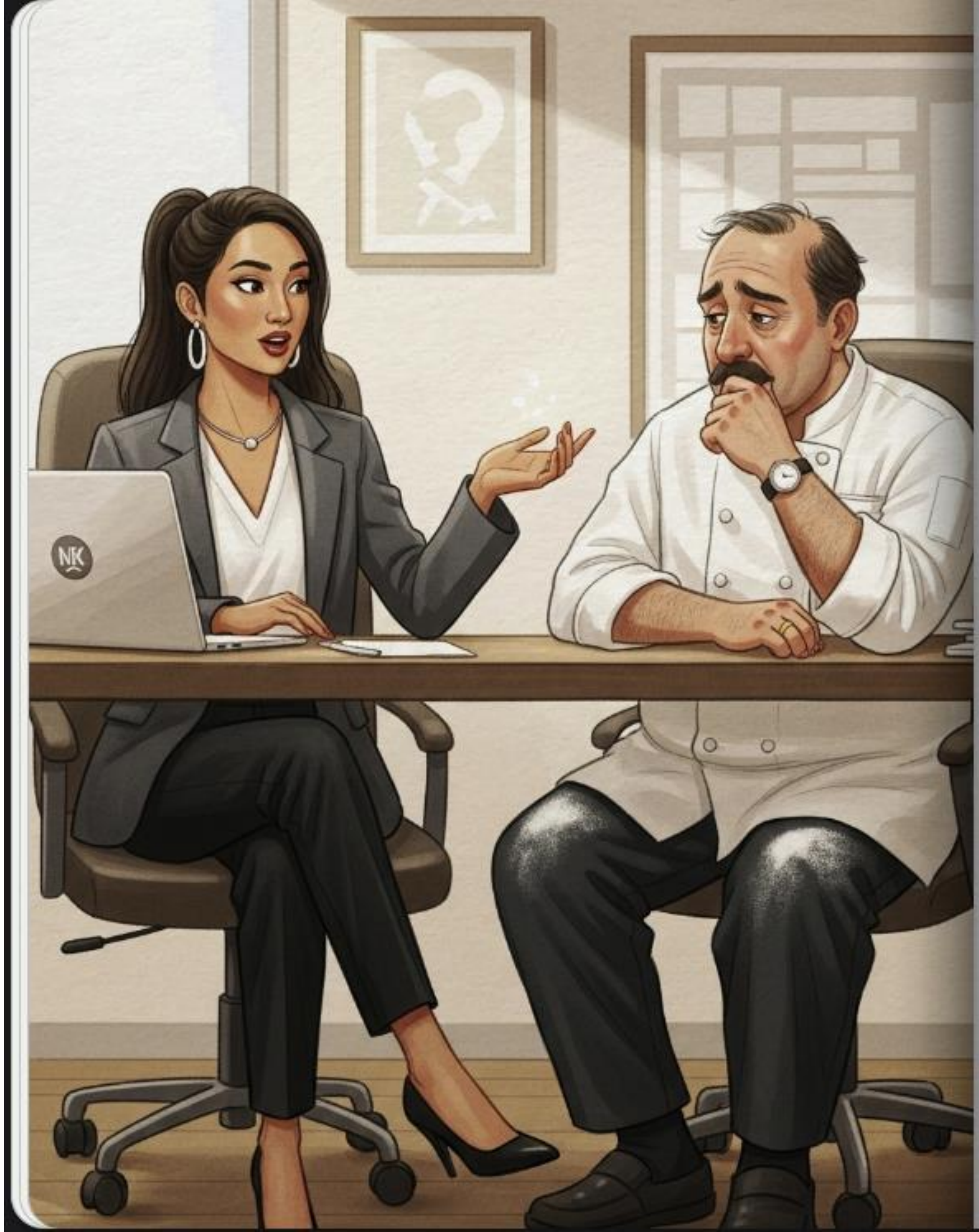
10 pizzas delivered!

"We're scaling delivery capacity like champions!"

Customer Service

2 complaints handled!

"Response times are excellent - customer satisfaction sorted!"



"You need to see the whole story. You need to use ThriveStack Full Journey Growth Intelligence."

"You must Measure your Growth,
Identify the Drivers, and Fix the Leaks."

The Universal Problem

This scenario plays out in countless businesses across every industry. Everyone has data, everyone has budget, but nobody has the story.

THE NEW WAY

Unified Growth Intelligence

Full Funnel

Connected

Full Journey Analytics

Seamlessly integrate marketing, product, revenue, and customer success data into a single source of truth.

Correlated

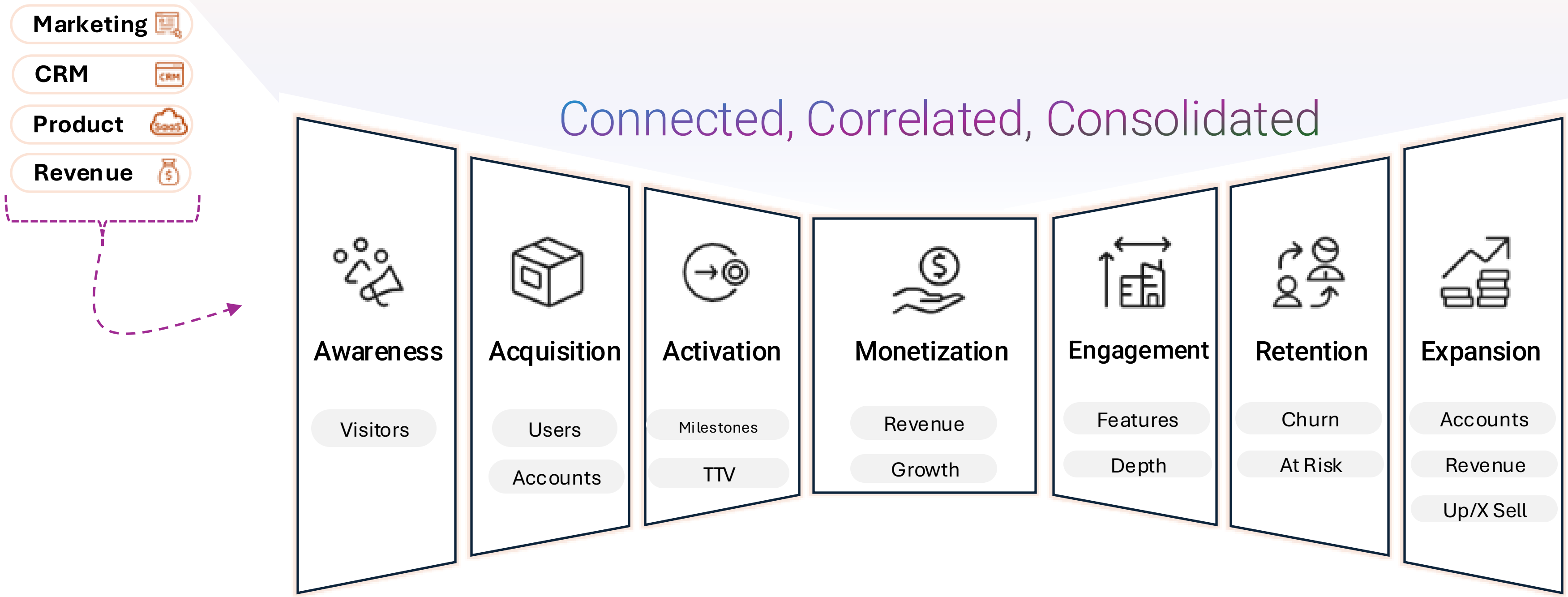
Growth Leaks Detector

250+ AI powered growth signals to automatically identify growth drivers and leaks.

Consolidated

Single platform for Growth

Eliminate data and tooling silos. Align all your teams with insights, customized views and scorecards



Marketing

CRM

Product

Revenue

Connected, Correlated, Consolidated

Awareness
- Visitors

Acquisition
- Users
- Accounts

Activation
- Milestones
- TTV

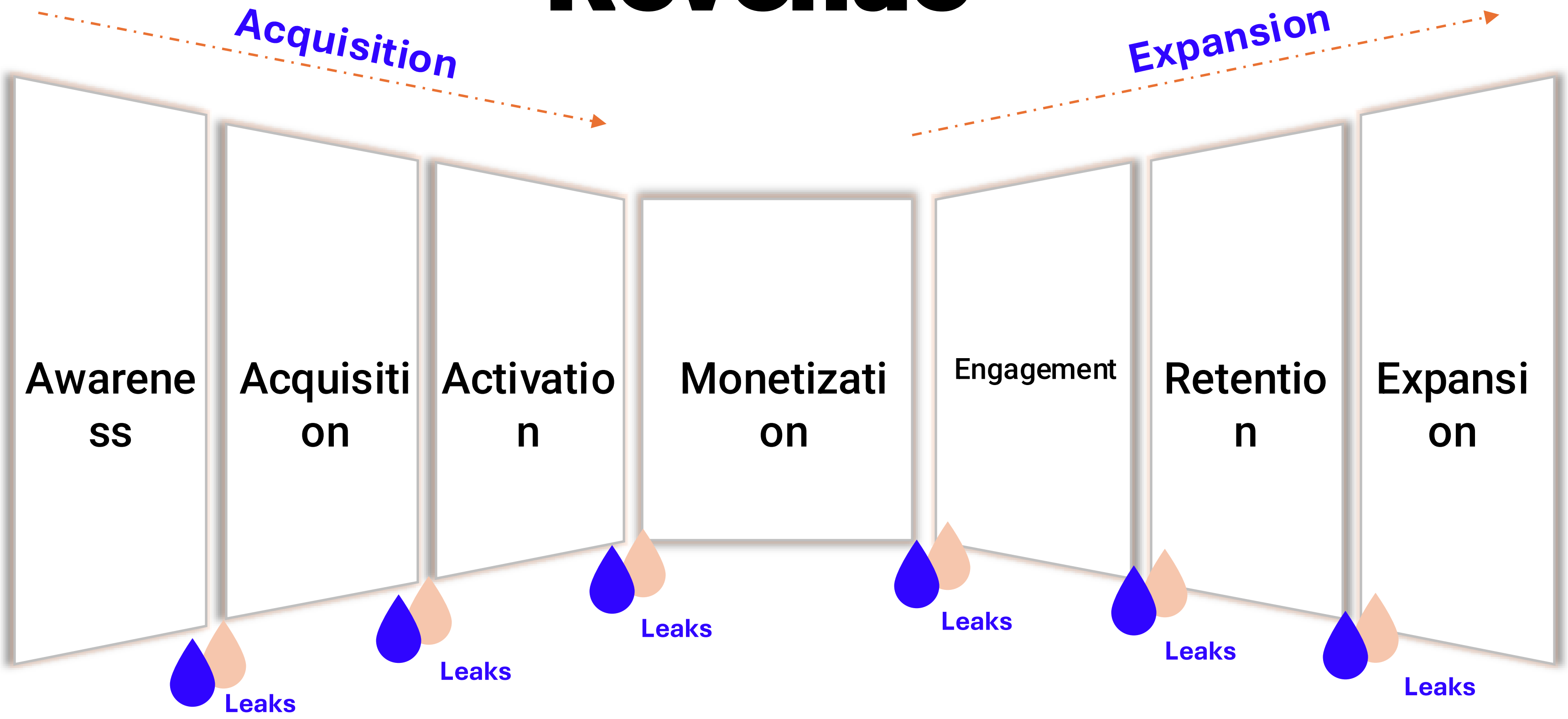
Monetization
- Revenue
- Growth

Engagement
- Features
- Depth

Retention
- Churn
- At Risk

Expansion
- Accounts
- Revenue
- Up/X Sell

Every Company leaks Customers and Revenue



The questions are...

Where?

How Much?

Why?

Can I stop them?

ThriveStack

How ThriveStack Chose ClickHouse

Parameters for Choosing OLAP:

1

Performance Fit

Query Latency, **Throughput Metrics,** Data Volume Handling

2

Cost/Pricing Fit

Predictable Costs, Operational Costs, Infrastructure Costs, Customer Scaling

3

Technical Architecture Fit

Stack Integration, Event Streaming, CDC Support, Batch Processing, Resource Isolation, Geographic Distribution

4

Customer Analytics Use-Cases Fit

Real-Time Customer Behavior, Customer Data Models, Cohort, Behavioral Segmentation, Customer Scores

Options We Considered OLAP:

1 ClickHouse

2 snowflake

3 PostgreSQL

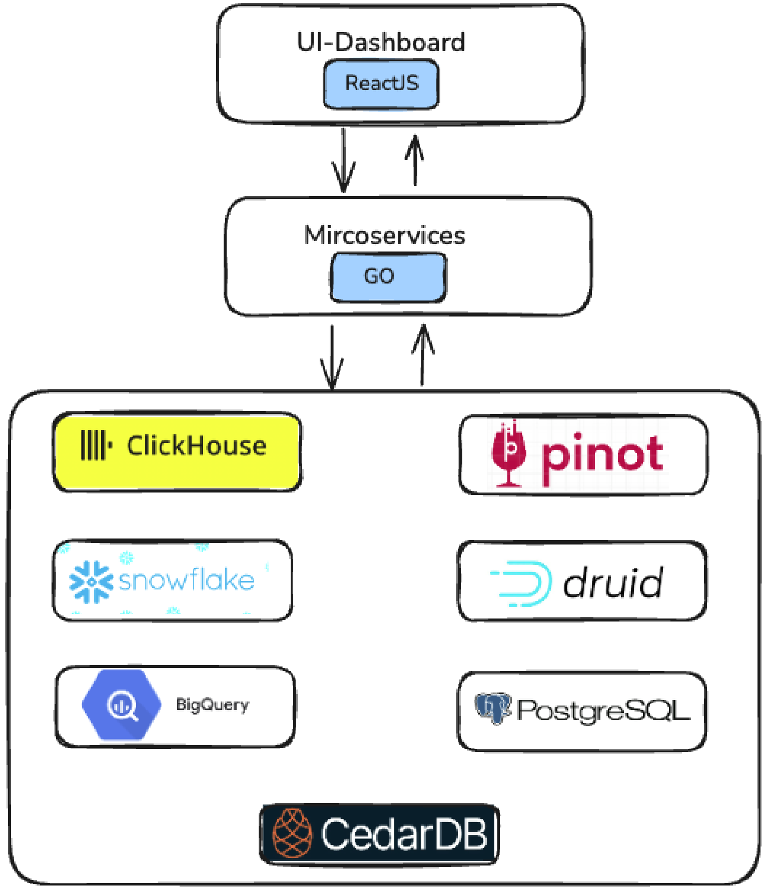
4 BigQuery

5 pinot

6 druid

7 CedarDB

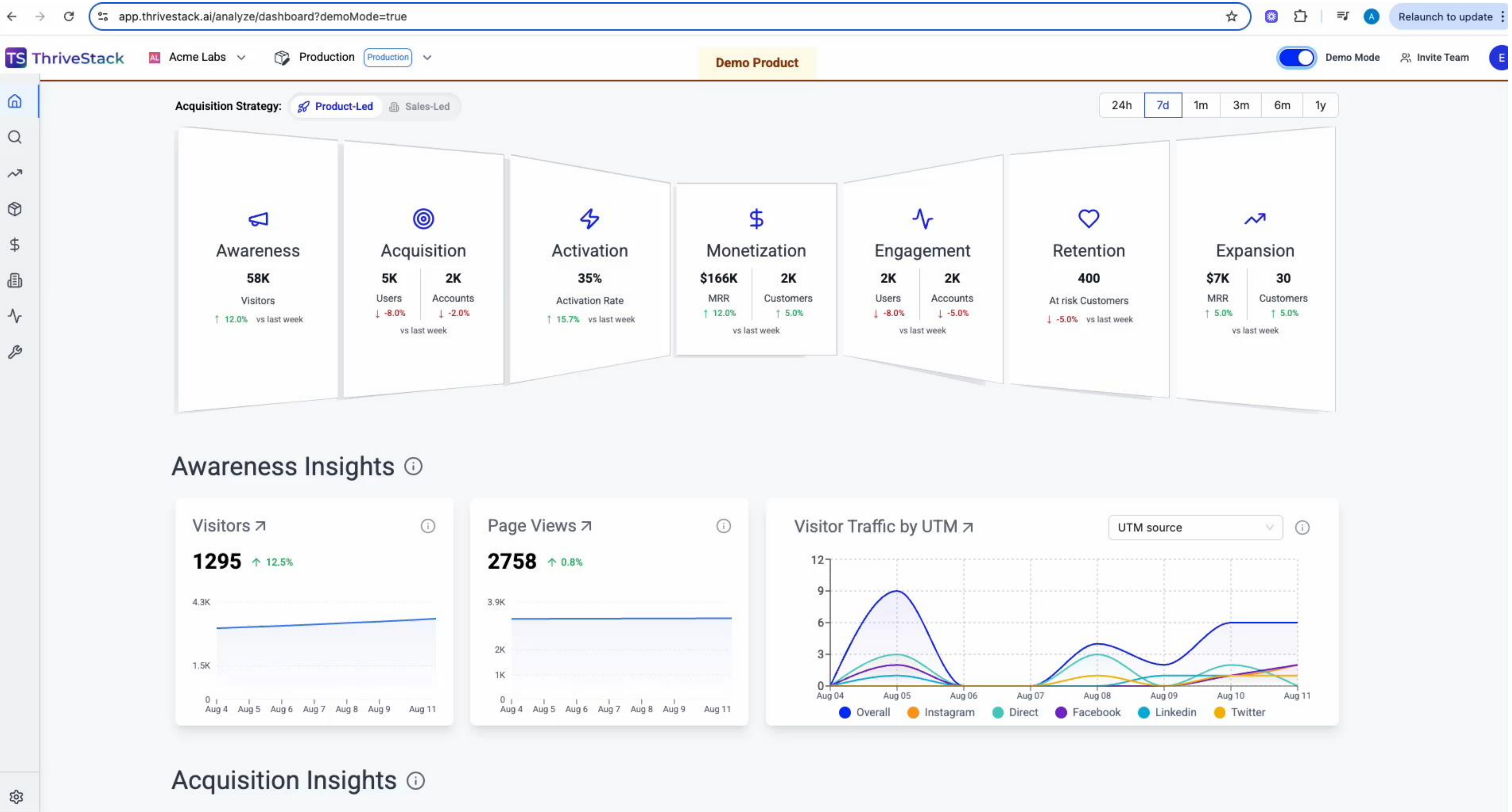
Test Setup



Total Metrics: 47 [Real Time + Score Computation + Correlation]

Data : 14 million rows + 12 tables + 12 GB Size

Use-Case & Metrics



app.thrivestack.ai/analyze/dashboard?demoMode=true

Awareness
58K Visitors
↑ 12.0% vs last week

Acquisition
5K Users ↓ -8.0%
2K Accounts ↓ -2.0%
vs last week

Activation
35% Activation Rate
↑ 15.7% vs last week

Monetization
$166K MRR ↑ 12.0%
2K Customers ↑ 5.0%
vs last week

Engagement
2K Users ↓ -8.0%
2K Accounts ↓ -5.0%
vs last week

Retention
400 At risk Customers
↓ -5.0% vs last week

Expansion
$7K MRR ↑ 5.0%
30 Customers ↑ 5.0%
vs last week

Awareness Insights

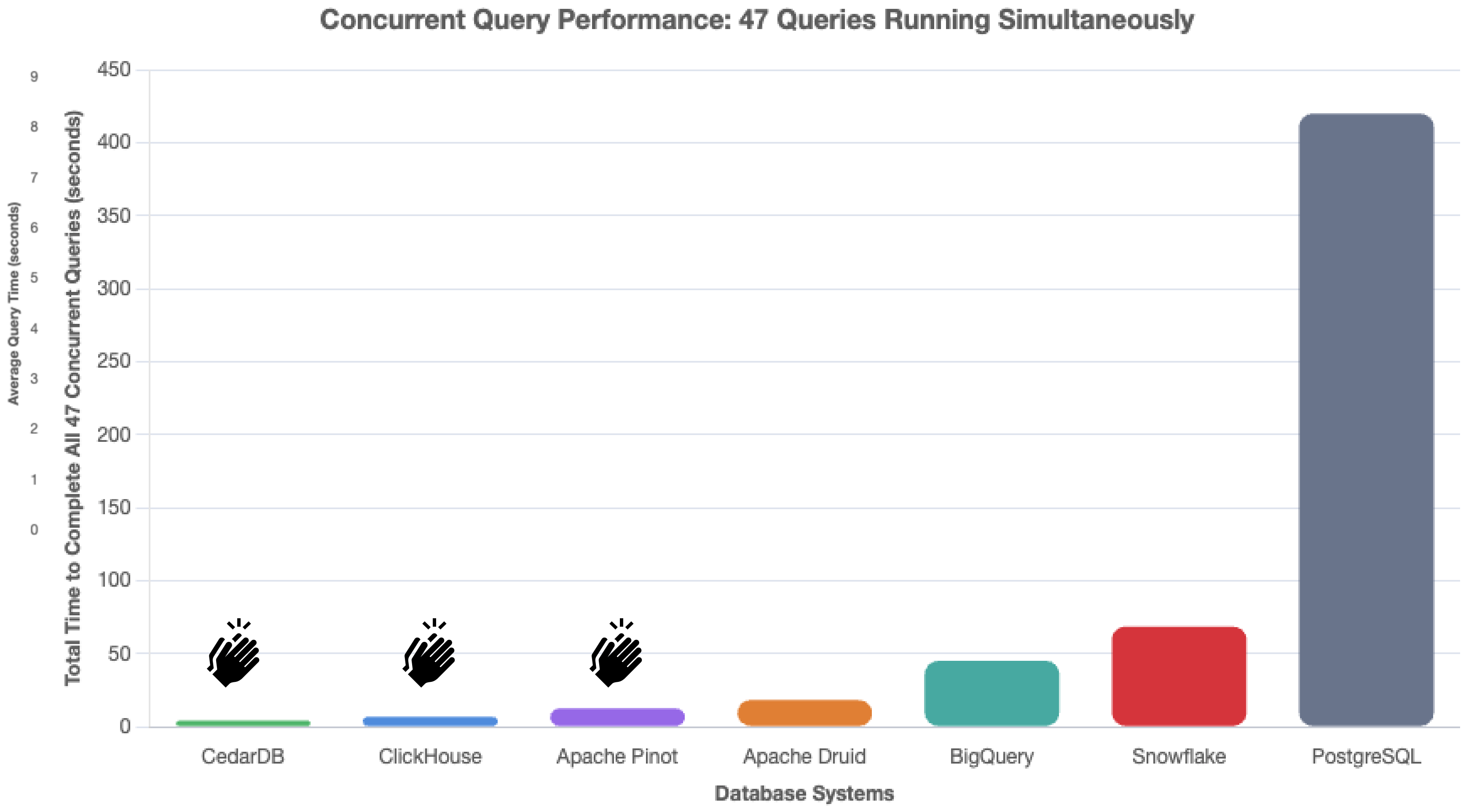
Visitors ↗
1295 ↑ 12.5%

Page Views ↗
2758 ↑ 0.8%

Visitor Traffic by UTM ↗
UTM source

Acquisition Insights

Performance Results



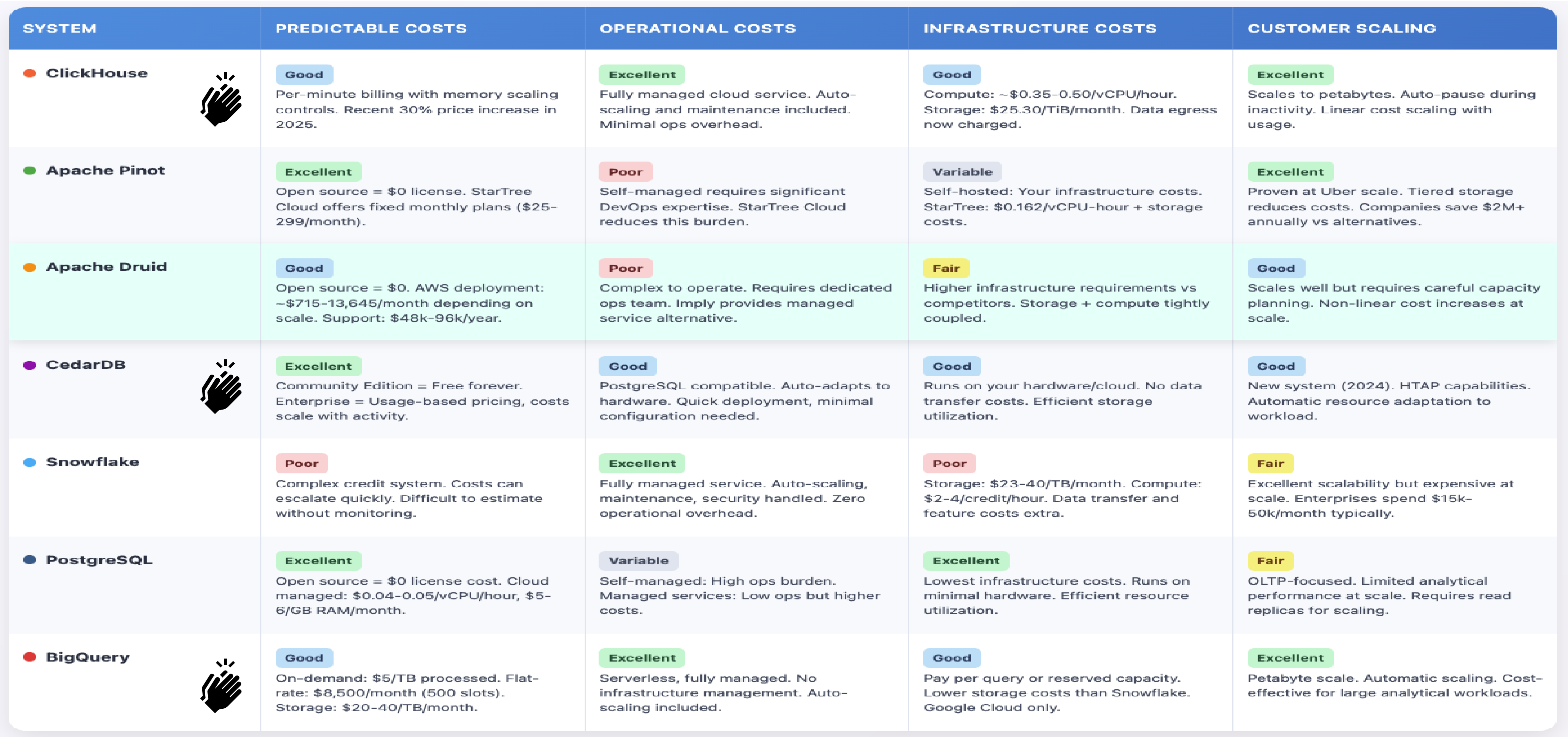
Database	Total Time for 47 Concurrent Queries
CedarDB	4.2s
ClickHouse	6.8s
Apache Pinot	12.5s
Apache Druid	18.3s
BigQuery	45.2s
Snowflake	1m 8s
PostgreSQL	7m 0s

https://benchmark.clickhouse.com

Total Metrics: 47 [Real Time + Score Computation + Correlation]

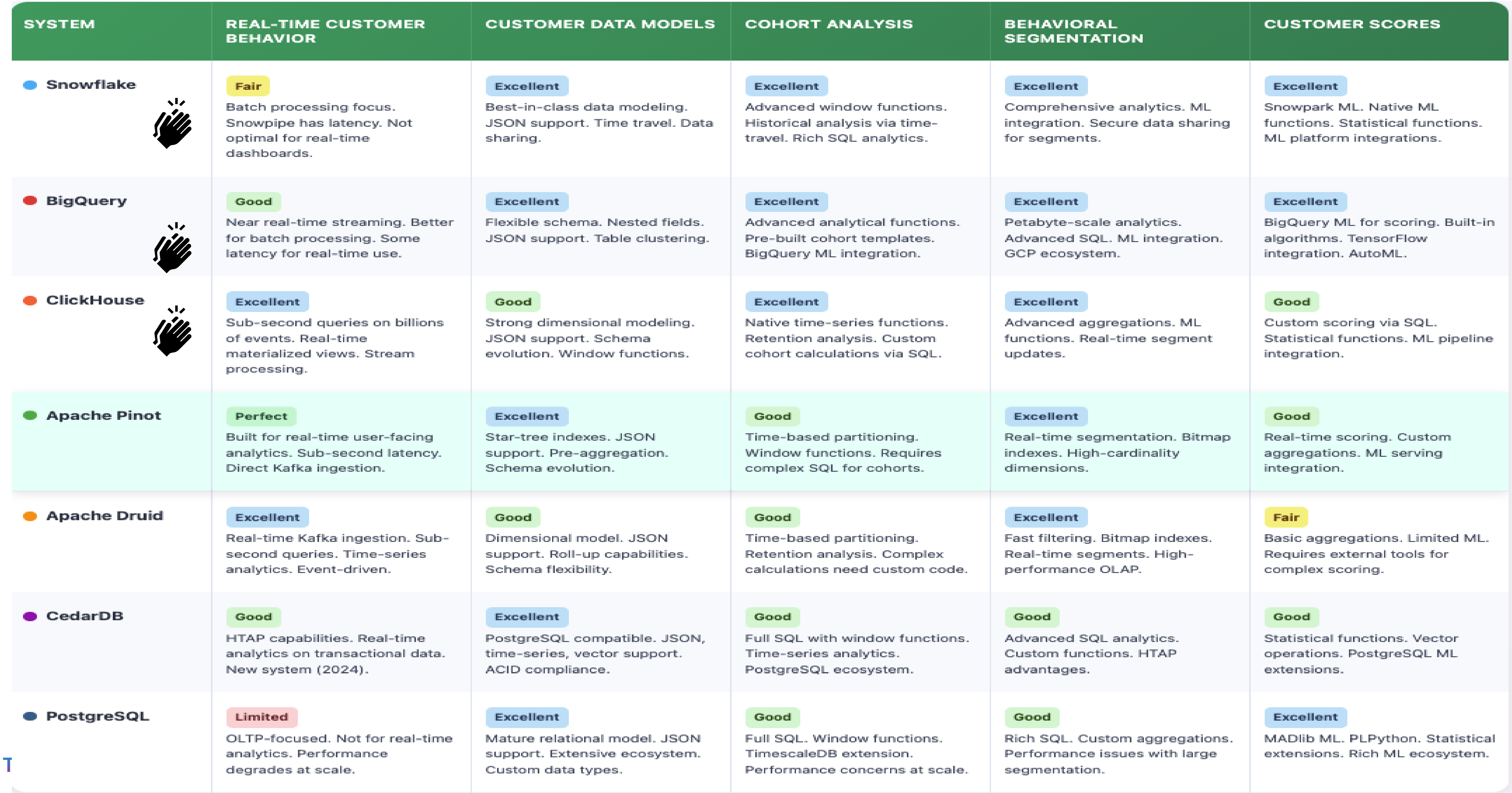
Data : 14 million rows + 12 tables + 12 GB Size

Cost Fit Results



SYSTEM	PREDICTABLE COSTS	OPERATIONAL COSTS	INFRASTRUCTURE COSTS	CUSTOMER SCALING
● ClickHouse 👏	Good Per-minute billing with memory scaling controls. Recent 30% price increase in 2025.	Excellent Fully managed cloud service. Auto-scaling and maintenance included. Minimal ops overhead.	Good Compute: ~$0.35-0.50/vCPU/hour. Storage: $25.30/TiB/month. Data egress now charged.	Excellent Scales to petabytes. Auto-pause during inactivity. Linear cost scaling with usage.
● Apache Pinot	Excellent Open source = $0 license. StarTree Cloud offers fixed monthly plans ($25-299/month).	Poor Self-managed requires significant DevOps expertise. StarTree Cloud reduces this burden.	Variable Self-hosted: Your infrastructure costs. StarTree: $0.162/vCPU-hour + storage costs.	Excellent Proven at Uber scale. Tiered storage reduces costs. Companies save $2M+ annually vs alternatives.
● Apache Druid	Good Open source = $0. AWS deployment: ~$715-13,645/month depending on scale. Support: $48k-96k/year.	Poor Complex to operate. Requires dedicated ops team. Imply provides managed service alternative.	Fair Higher infrastructure requirements vs competitors. Storage + compute tightly coupled.	Good Scales well but requires careful capacity planning. Non-linear cost increases at scale.
● CedarDB 👏	Excellent Community Edition = Free forever. Enterprise = Usage-based pricing, costs scale with activity.	Good PostgreSQL compatible. Auto-adapts to hardware. Quick deployment, minimal configuration needed.	Good Runs on your hardware/cloud. No data transfer costs. Efficient storage utilization.	Good New system (2024). HTAP capabilities. Automatic resource adaptation to workload.
● Snowflake	Poor Complex credit system. Costs can escalate quickly. Difficult to estimate without monitoring.	Excellent Fully managed service. Auto-scaling, maintenance, security handled. Zero operational overhead.	Poor Storage: $23-40/TB/month. Compute: $2-4/credit/hour. Data transfer and feature costs extra.	Fair Excellent scalability but expensive at scale. Enterprises spend $15k-50k/month typically.
● PostgreSQL	Excellent Open source = $0 license cost. Cloud managed: $0.04-0.05/vCPU/hour, $5-6/GB RAM/month.	Variable Self-managed: High ops burden. Managed services: Low ops but higher costs.	Excellent Lowest infrastructure costs. Runs on minimal hardware. Efficient resource utilization.	Fair OLTP-focused. Limited analytical performance at scale. Requires read replicas for scaling.
● BigQuery 👏	Good On-demand: $5/TB processed. Flat-rate: $8,500/month (500 slots). Storage: $20-40/TB/month.	Excellent Serverless, fully managed. No infrastructure management. Auto-scaling included.	Good Pay per query or reserved capacity. Lower storage costs than Snowflake. Google Cloud only.	Excellent Petabyte scale. Automatic scaling. Cost-effective for large analytical workloads.

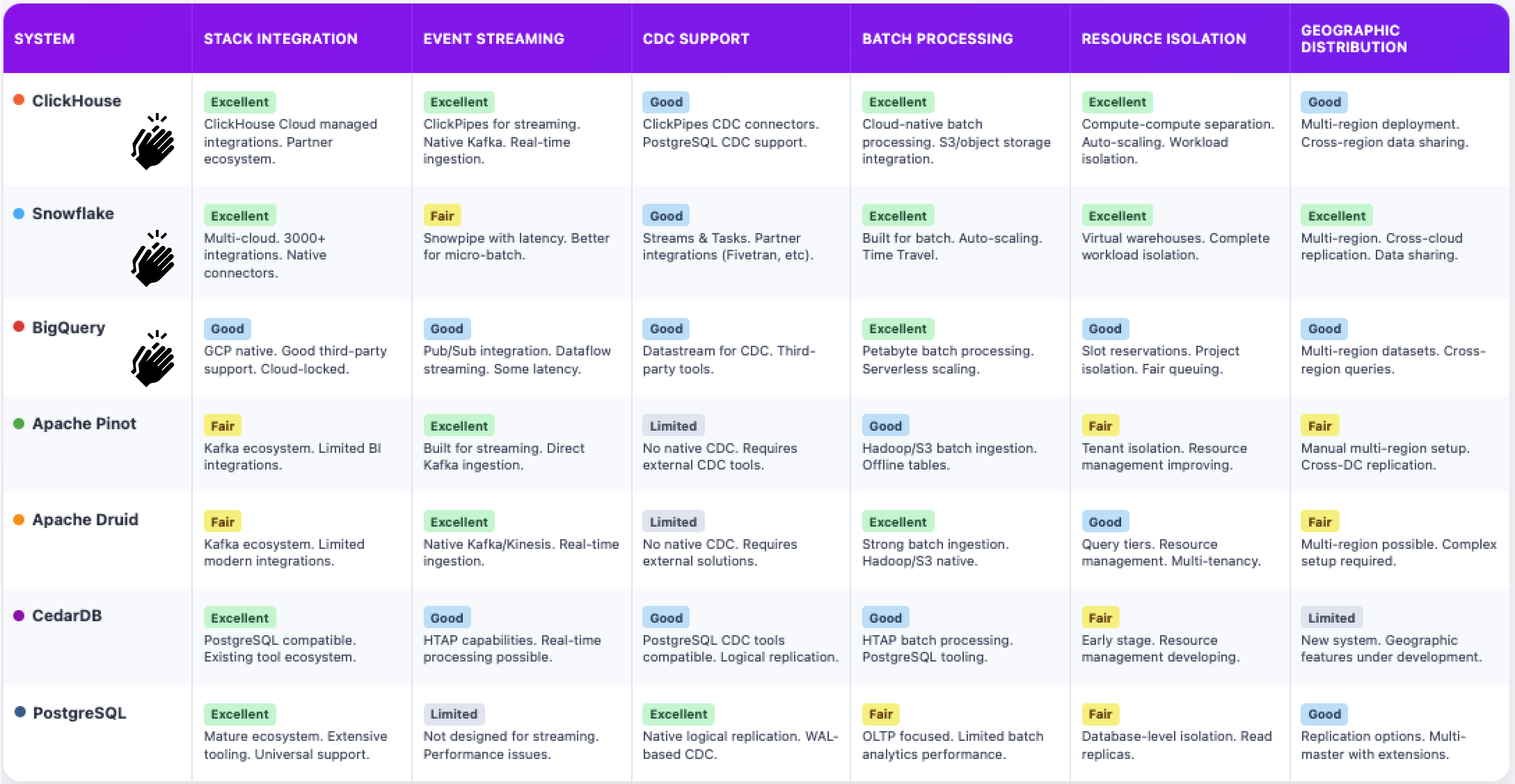
Business Use-case Fit



SYSTEM	REAL-TIME CUSTOMER BEHAVIOR	CUSTOMER DATA MODELS	COHORT ANALYSIS	BEHAVIORAL SEGMENTATION	CUSTOMER SCORES
● Snowflake 👏	Fair Batch processing focus. Snowpipe has latency. Not optimal for real-time dashboards.	Excellent Best-in-class data modeling. JSON support. Time travel. Data sharing.	Excellent Advanced window functions. Historical analysis via time-travel. Rich SQL analytics.	Excellent Comprehensive analytics. ML integration. Secure data sharing for segments.	Excellent Snowpark ML. Native ML functions. Statistical functions. ML platform integrations.
● BigQuery 👏	Good Near real-time streaming. Better for batch processing. Some latency for real-time use.	Excellent Flexible schema. Nested fields. JSON support. Table clustering.	Excellent Advanced analytical functions. Pre-built cohort templates. BigQuery ML integration.	Excellent Petabyte-scale analytics. Advanced SQL. ML integration. GCP ecosystem.	Excellent BigQuery ML for scoring. Built-in algorithms. TensorFlow integration. AutoML.
● ClickHouse 👏	Excellent Sub-second queries on billions of events. Real-time materialized views. Stream processing.	Good Strong dimensional modeling. JSON support. Schema evolution. Window functions.	Excellent Native time-series functions. Retention analysis. Custom cohort calculations via SQL.	Excellent Advanced aggregations. ML functions. Real-time segment updates.	Good Custom scoring via SQL. Statistical functions. ML pipeline integration.
● Apache Pinot	Perfect Built for real-time user-facing analytics. Sub-second latency. Direct Kafka ingestion.	Excellent Star-tree indexes. JSON support. Pre-aggregation. Schema evolution.	Good Time-based partitioning. Window functions. Requires complex SQL for cohorts.	Excellent Real-time segmentation. Bitmap indexes. High-cardinality dimensions.	Good Real-time scoring. Custom aggregations. ML serving integration.
● Apache Druid	Excellent Real-time Kafka ingestion. Sub-second queries. Time-series analytics. Event-driven.	Good Dimensional model. JSON support. Roll-up capabilities. Schema flexibility.	Good Time-based partitioning. Retention analysis. Complex calculations need custom code.	Excellent Fast filtering. Bitmap indexes. Real-time segments. High-performance OLAP.	Fair Basic aggregations. Limited ML. Requires external tools for complex scoring.
● CedarDB	Good HTAP capabilities. Real-time analytics on transactional data. New system (2024).	Excellent PostgreSQL compatible. JSON, time-series, vector support. ACID compliance.	Good Full SQL with window functions. Time-series analytics. PostgreSQL ecosystem.	Good Advanced SQL analytics. Custom functions. HTAP advantages.	Good Statistical functions. Vector operations. PostgreSQL ML extensions.
● PostgreSQL	Limited OLTP-focused. Not for real-time analytics. Performance degrades at scale.	Excellent Mature relational model. JSON support. Extensive ecosystem. Custom data types.	Good Full SQL. Window functions. TimescaleDB extension. Performance concerns at scale.	Good Rich SQL. Custom aggregations. Performance issues with large segmentation.	Excellent MADlib ML. PLPython. Statistical extensions. Rich ML ecosystem.

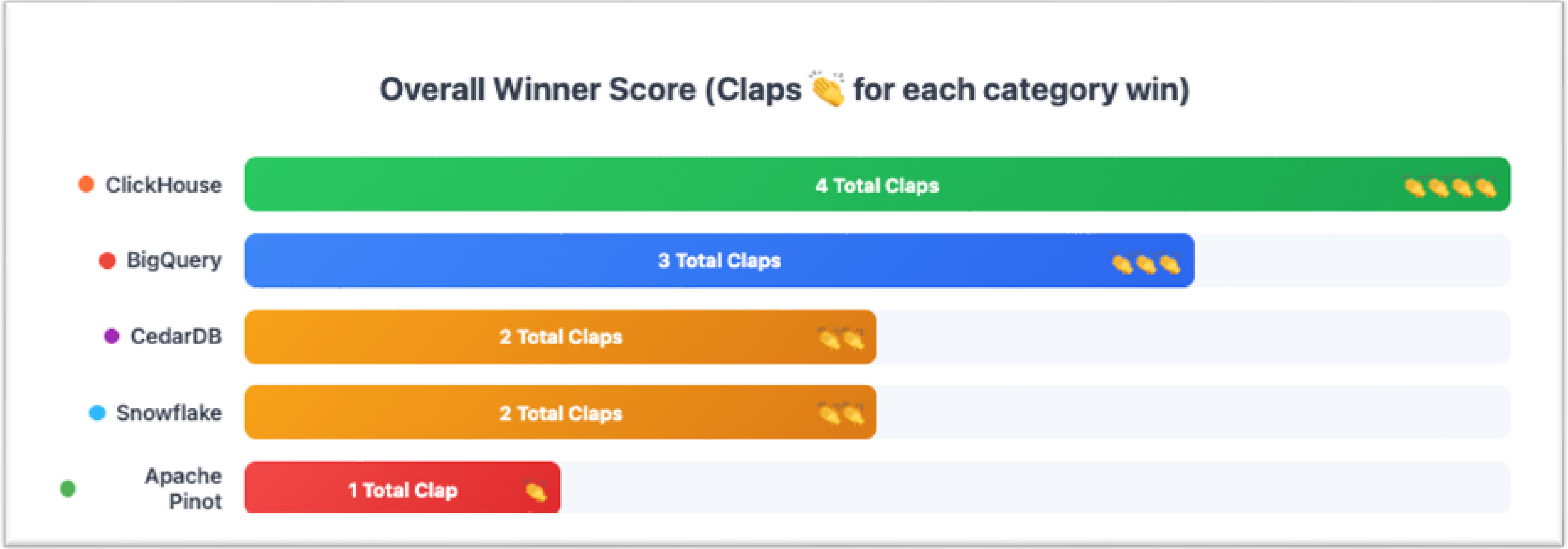
T

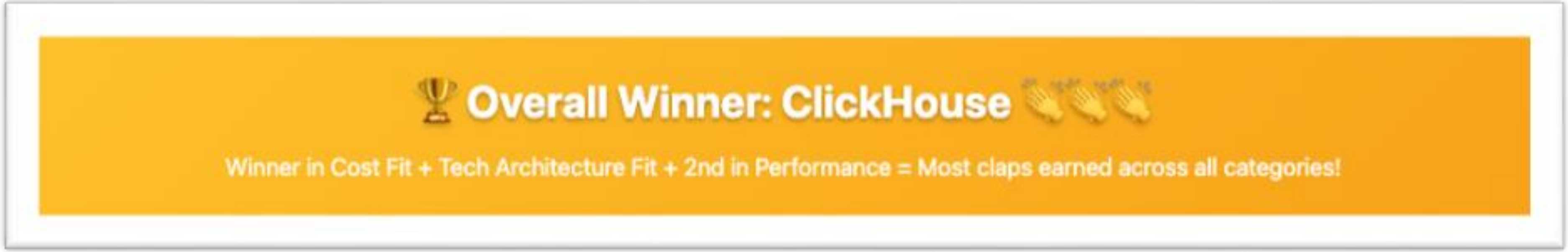
Technical Architecture Fit



SYSTEM	STACK INTEGRATION	EVENT STREAMING	CDC SUPPORT	BATCH PROCESSING	RESOURCE ISOLATION	GEOGRAPHIC DISTRIBUTION
● ClickHouse 👏	Excellent ClickHouse Cloud managed integrations. Partner ecosystem.	Excellent ClickPipes for streaming. Native Kafka. Real-time ingestion.	Good ClickPipes CDC connectors. PostgreSQL CDC support.	Excellent Cloud-native batch processing. S3/object storage integration.	Excellent Compute-compute separation. Auto-scaling. Workload isolation.	Good Multi-region deployment. Cross-region data sharing.
● Snowflake 👏	Excellent Multi-cloud. 3000+ integrations. Native connectors.	Fair Snowpipe with latency. Better for micro-batch.	Good Streams & Tasks. Partner integrations (Fivetran, etc).	Excellent Built for batch. Auto-scaling. Time Travel.	Excellent Virtual warehouses. Complete workload isolation.	Excellent Multi-region. Cross-cloud replication. Data sharing.
● BigQuery 👏	Good GCP native. Good third-party support. Cloud-locked.	Good Pub/Sub integration. Dataflow streaming. Some latency.	Good Datastream for CDC. Third-party tools.	Excellent Petabyte batch processing. Serverless scaling.	Good Slot reservations. Project isolation. Fair queuing.	Good Multi-region datasets. Cross-region queries.
● Apache Pinot	Fair Kafka ecosystem. Limited BI integrations.	Excellent Built for streaming. Direct Kafka ingestion.	Limited No native CDC. Requires external CDC tools.	Good Hadoop/S3 batch ingestion. Offline tables.	Fair Tenant isolation. Resource management improving.	Fair Manual multi-region setup. Cross-DC replication.
● Apache Druid	Fair Kafka ecosystem. Limited modern integrations.	Excellent Native Kafka/Kinesis. Real-time ingestion.	Limited No native CDC. Requires external solutions.	Excellent Strong batch ingestion. Hadoop/S3 native.	Good Query tiers. Resource management. Multi-tenancy.	Fair Multi-region possible. Complex setup required.
● CedarDB	Excellent PostgreSQL compatible. Existing tool ecosystem.	Good HTAP capabilities. Real-time processing possible.	Good PostgreSQL CDC tools compatible. Logical replication.	Good HTAP batch processing. PostgreSQL tooling.	Fair Early stage. Resource management developing.	Limited New system. Geographic features under development.
● PostgreSQL	Excellent Mature ecosystem. Extensive tooling. Universal support.	Limited Not designed for streaming. Performance issues.	Excellent Native logical replication. WAL-based CDC.	Fair OLTP focused. Limited batch analytics performance.	Fair Database-level isolation. Read replicas.	Good Replication options. Multi-master with extensions.

Winner





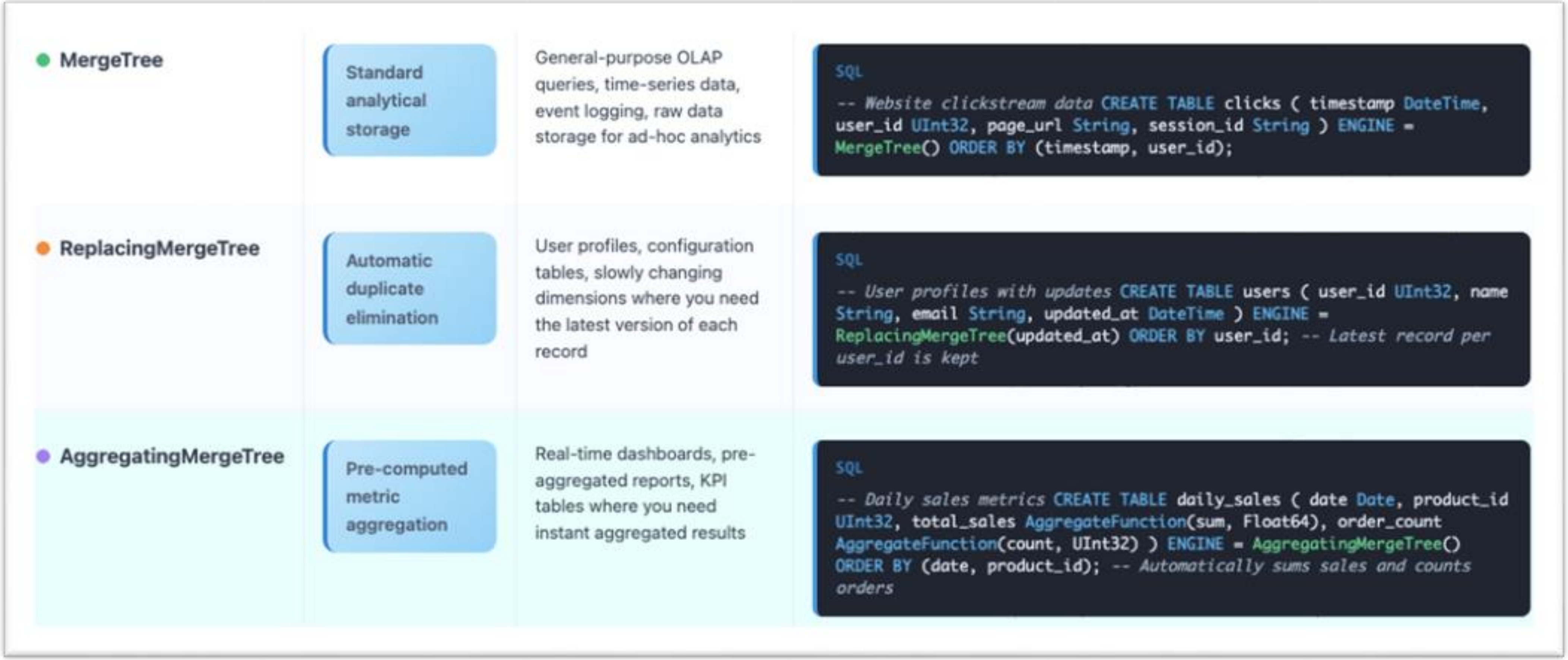
🏆 Overall Winner: ClickHouse 👏👏👏

Winner in Cost Fit + Tech Architecture Fit + 2nd in Performance = Most claps earned across all categories!

What We Learnt About Effective Use Of ClickHouse

Effective Use Of Table engines

1
2
3
4



● MergeTree	Standard analytical storage	General-purpose OLAP queries, time-series data, event logging, raw data storage for ad-hoc analytics	SQL `-- Website clickstream data CREATE TABLE clicks ( timestamp DateTime, user_id UInt32, page_url String, session_id String ) ENGINE = MergeTree() ORDER BY (timestamp, user_id);`
● ReplacingMergeTree	Automatic duplicate elimination	User profiles, configuration tables, slowly changing dimensions where you need the latest version of each record	SQL `-- User profiles with updates CREATE TABLE users ( user_id UInt32, name String, email String, updated_at DateTime ) ENGINE = ReplacingMergeTree(updated_at) ORDER BY user_id; -- Latest record per user_id is kept`
● AggregatingMergeTree	Pre-computed metric aggregation	Real-time dashboards, pre-aggregated reports, KPI tables where you need instant aggregated results	SQL `-- Daily sales metrics CREATE TABLE daily_sales ( date Date, product_id UInt32, total_sales AggregateFunction(sum, Float64), order_count AggregateFunction(count, UInt32) ) ENGINE = AggregatingMergeTree() ORDER BY (date, product_id); -- Automatically sums sales and counts orders`

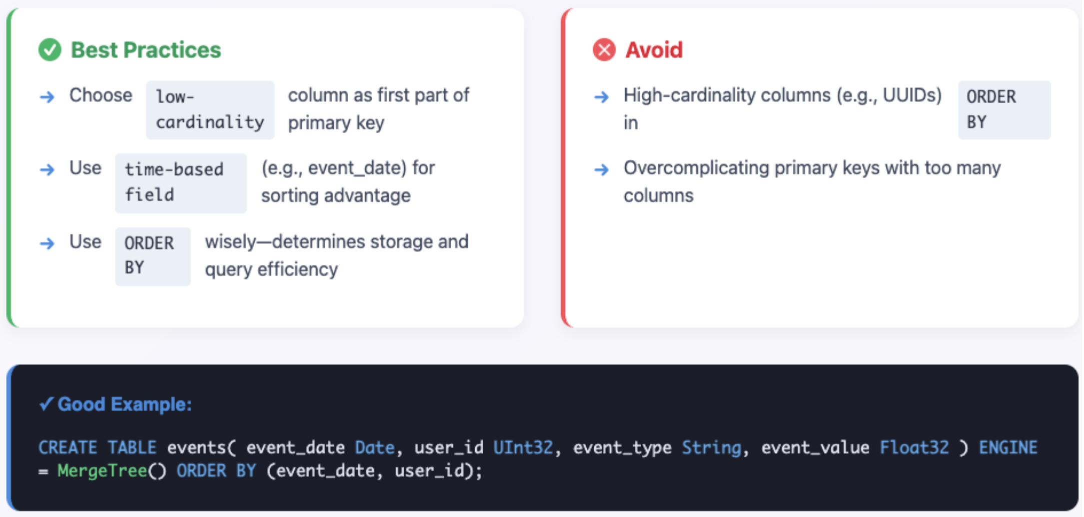
Optimize Primary Key and Partitioning

1

2

3

4



✓ Best Practices

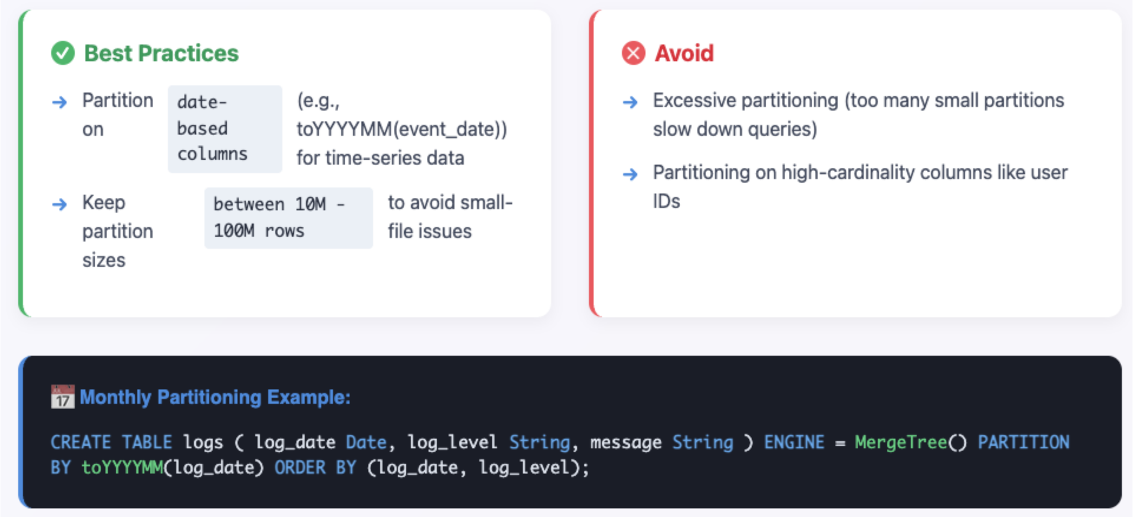
- → Choose `low-cardinality` column as first part of primary key
- → Use `time-based field` (e.g., event_date) for sorting advantage
- → Use `ORDER BY` wisely—determines storage and query efficiency

☒ Avoid

- → High-cardinality columns (e.g., UUIDs) in `ORDER BY`
- → Overcomplicating primary keys with too many columns

✓ Good Example:

```
CREATE TABLE events( event_date Date, user_id UInt32, event_type String, event_value Float32 ) ENGINE
= MergeTree() ORDER BY (event_date, user_id);
```



✓ Best Practices

- → Partition on `date-based columns` (e.g., toYYYYMM(event_date)) for time-series data
- → Keep partition sizes `between 10M - 100M rows` to avoid small-file issues

☒ Avoid

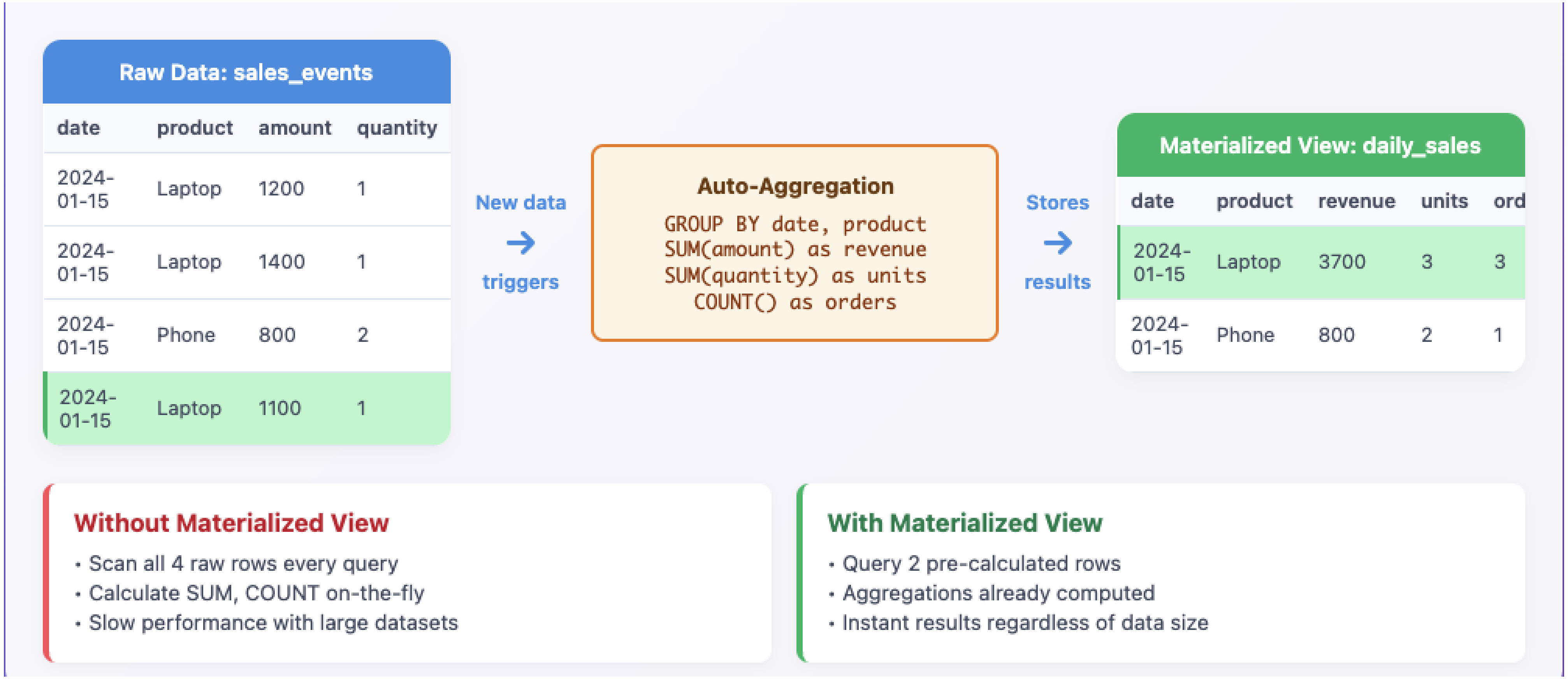
- → Excessive partitioning (too many small partitions slow down queries)
- → Partitioning on high-cardinality columns like user IDs

📅 Monthly Partitioning Example:

```
CREATE TABLE logs ( log_date Date, log_level String, message String ) ENGINE = MergeTree() PARTITION
BY toYYYYMM(log_date) ORDER BY (log_date, log_level);
```

Precompute Aggregations using MV

1
2
3
4



Raw Data: sales_events

date	product	amount	quantity
2024-01-15	Laptop	1200	1
2024-01-15	Laptop	1400	1
2024-01-15	Phone	800	2
2024-01-15	Laptop	1100	1

New data → triggers

Auto-Aggregation

```
GROUP BY date, product
SUM(amount) as revenue
SUM(quantity) as units
COUNT() as orders
```

Stores → results

Materialized View: daily_sales

date	product	revenue	units	ord
2024-01-15	Laptop	3700	3	3
2024-01-15	Phone	800	2	1

Without Materialized View

- Scan all 4 raw rows every query
- Calculate SUM, COUNT on-the-fly
- Slow performance with large datasets

With Materialized View

- Query 2 pre-calculated rows
- Aggregations already computed
- Instant results regardless of data size

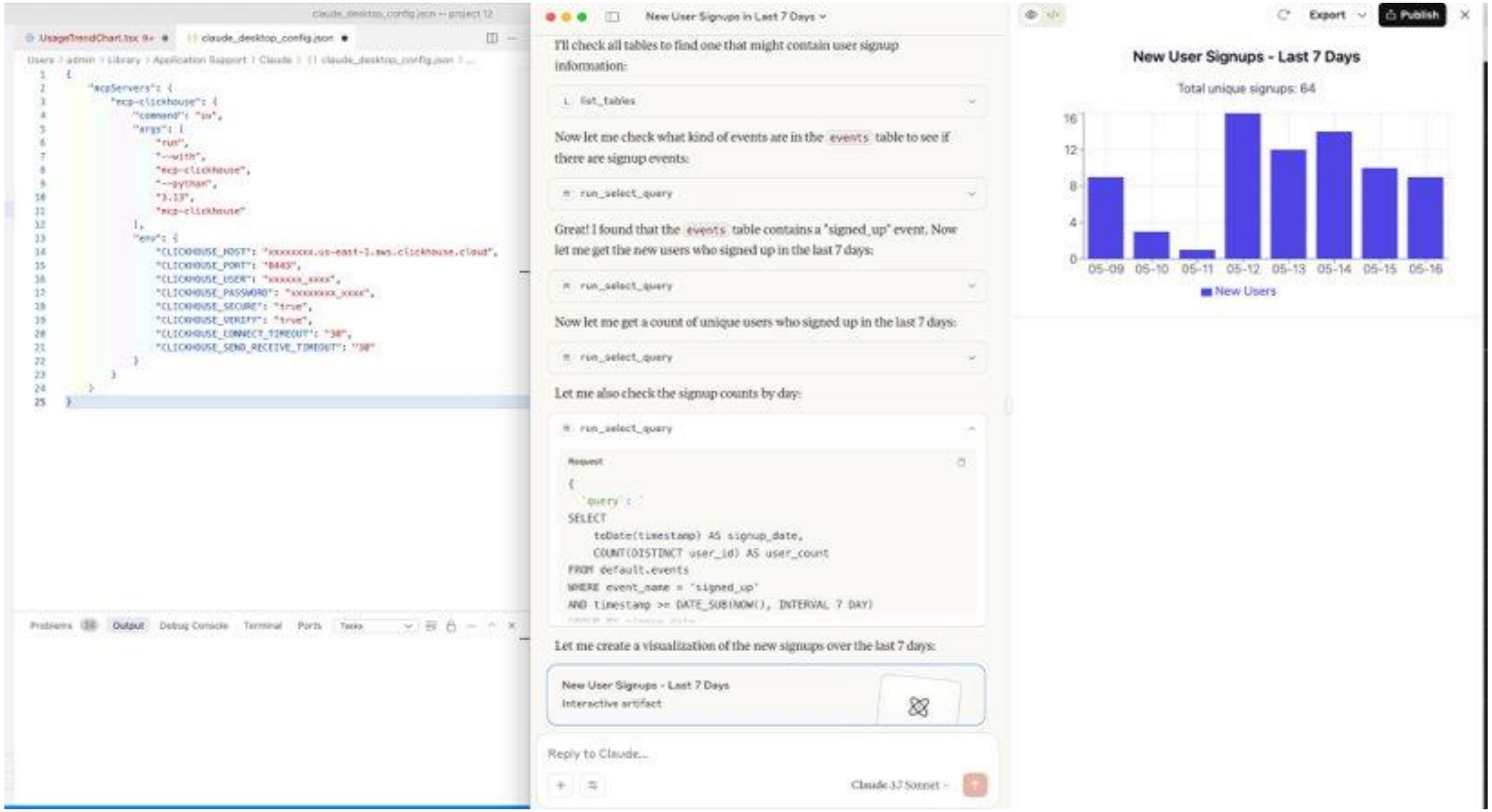
ClickHouse MCP Integrations

1

2

3

4


claude_desktop_config.json — project 12
UsageTrendChart.tsx 9+
claude_desktop_config.json
Users › admin › Library › Application Support › Claude › claude_desktop_config.json › ...
Problems
Output
Debug Console
Terminal
Ports
Tasks
New User Signups in Last 7 Days
Export
Publish
16
12
8
4
0
05-09
05-10
05-11
05-12
05-13
05-14
05-15
05-16
New Users

```
{
    "mcpServers": {
        "mcp-clickhouse": {
            "command": "uv",
            "args": [
                "run",
                "--with",
                "mcp-clickhouse",
                "--python",
                "3.13",
                "mcp-clickhouse"
            ],
            "env": {
                "CLICKHOUSE_HOST": "xxxxxxxx.us-east-1.aws.clickhouse.cloud",
                "CLICKHOUSE_PORT": "8443",
                "CLICKHOUSE_USER": "xxxxxx_xxxx",
                "CLICKHOUSE_PASSWORD": "xxxxxxxx_xxxx",
                "CLICKHOUSE_SECURE": "true",
                "CLICKHOUSE_VERIFY": "true",
                "CLICKHOUSE_CONNECT_TIMEOUT": "30",
                "CLICKHOUSE_SEND_RECEIVE_TIMEOUT": "30"
            }
        }
    }
}
```

I'll check all tables to find one that might contain user signup information:

list_tables

Now let me check what kind of events are in the `events` table to see if there are signup events:

run_select_query

Great! I found that the `events` table contains a "signed_up" event. Now let me get the new users who signed up in the last 7 days:

run_select_query

Now let me get a count of unique users who signed up in the last 7 days:

run_select_query

Let me also check the signup counts by day:

run_select_query

Request

```
{
  'query': '
SELECT
    toDate(timestamp) AS signup_date,
    COUNT(DISTINCT user_id) AS user_count
FROM default.events
WHERE event_name = 'signed_up'
AND timestamp >= DATE_SUB(NOW(), INTERVAL 7 DAY)
```

Let me create a visualization of the new signups over the last 7 days:

New User Signups - Last 7 Days
Interactive artifact

Reply to Claude...

Claude 3.7 Sonnet

New User Signups - Last 7 Days

Total unique signups: 64

The Results...

PRODUCT DEVELOPMENT

20% Reduction

in Engineering Resources

PRODUCT EXPERIMENTATION

3x faster

in release velocity

Develop → Release → Feedback

COGS COSTS

40% higher

More engineering resources now use ClickHouse as opposed to only Backend / Data engineers

Thank you

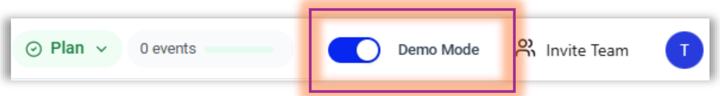
Give it a spin

Experience it yourself, at your own pace

Signup & try

https://www.thrivestack.ai/

Try the demo mode
After you signup, try the demo mode at the top of the screen



Schedule a demo/Onboarding

Bring in your team for a Demo/product onboarding

- Learn how to get the best out of the product
- Customize it to the stage of the company

Schedule a call