Data Platforms — Digest

Definition

A data platform is the infrastructure and set of tools an organization uses to collect, store, process, and serve data so applications and people can reliably use it. It's the plumbing that moves data from where it's generated to where it's needed, in usable form.

Real-life usage

- E-commerce: clickstream, orders, and inventory land in a warehouse (Snowflake/BigQuery/Redshift) and feed both BI dashboards and ML models (recommendations, fraud detection) from the same data.
- Fintech: transaction data flows via streaming pipelines (Kafka) into systems supporting both real-time fraud scoring and end-of-day regulatory reporting.
- Healthcare: patient records, device telemetry, and claims data are unified under strict governance (HIPAA) while remaining queryable for research.
- SaaS internal analytics: app DB → ETL/ELT → warehouse → dashboards for product/finance teams.
- Analogy: a city utility grid — treatment plant (ingestion/cleaning), pipes/reservoirs (storage), pumping stations (processing), taps (serving). Break any layer, everything downstream is affected.

5 Whys — Root Cause

- Why do platforms exist? Raw source systems are wrong for two independent reasons.
- **Reason 1 — contention:** OLTP DBs are optimized for row-based point lookups. Large analytical scans evict the transactional working set from buffer cache and hold locks longer, causing cache misses and slow transactional reads (e.g., checkout).
- Read replicas solve contention but not query *shape* — row-store/3NF schemas are structurally bad at large scans/aggregations, hence column-store, star schemas, and pre-aggregation.
- **Reason 2 — incoherence:** each source system encodes its own implicit business logic (e.g., three different definitions of "active customer" across CRM, billing, and app DB).

With N sources and M consumers and no shared contract, definitions drift — it's structurally guaranteed, not a one-off mistake.
- Platforms fix this by centralizing logic into governed, computed-once definitions everyone reads from.

Value vs. Risk

- **Value:** understanding the "why" lets you choose stack components by the guarantee a workload needs (ordering, replayability, latency, managed vs. self-hosted) instead of by popularity or familiarity — avoiding redundant tools solving the same problem.
- **Risk — performance/consistency:** black-boxing the platform reproduces the original problems (contention, definitional drift) plus new complexity/understanding debt (unclear why each tool is in the stack).
- **Risk — trust erosion:** when dashboards disagree and no one can explain why, people stop trusting the platform and quietly bypass it for gut-feel or spreadsheets — a costly, hard-to-reverse adoption failure.

Revision #2
Created 31 August 2026 08:37:58 by N8N
Updated 28 September 2026 14:39:06 by EMB