Modern Data Engineering – End■to■End System Design Guide

Single Source of Truth for Interviews and Real■World Platform Design

1. Design Philosophy

• Design for scale, failure, and change from day one.
• Prefer immutable raw data and deterministic transformations.
• Optimize for reprocessing, observability, and cost efficiency.
• Treat data, schema, and quality rules as versioned code.

2. Reference Architecture

• Kappa■first architecture: streaming as the default, batch as bounded streaming.
• Medallion pattern: Bronze (raw), Silver (conformed), Gold (serving).
• Separation of concerns: ingestion ≠ transformation ≠ serving.

3. Data Ingestion – Source Patterns

File (CSV/JSON): incremental discovery, schema enforcement, quarantine bad records.
RDBMS: CDC preferred; otherwise incremental pulls with partitioned reads.
Kafka/Event Streams: event■time processing, watermarking, deduplication.

4. Scalability at Ingestion

• Parallelism: partitions, file splits, micro■batch sizing.
• Backpressure: rate limiting, autoscaling, queue depth monitoring.
• Avoid small files using compaction and correct partition design.

5. Schema Management & Data Contracts

• Versioned schemas with backward compatibility rules.
• Explicit nullability, domains, and semantics.
• Machine■readable contracts used in CI/CD and runtime checks.

6. Watermarking & Late Data Handling

• Always separate event_time, ingest_time, processing_time.
• Define late arrival tolerance per dataset.
• Watermarks bound state and ensure deterministic aggregates.

7. Storage Design

• Columnar formats with ACID support (Delta/Iceberg/Hudi).
• Partition by low■cardinality, query■driven keys.
• Use clustering/Z■ordering for high■cardinality filters.
• Enforce file size standards and compaction.

8. Transformations – Mandatory Set

• Type standardization and timezone normalization.
• Null handling with explicit business meaning.
• Deduplication using deterministic keys.
• Reference data enrichment.
• CDC merge logic and SCD Type 1/2 handling.

9. CDC & SCD Design

• Capture all change events in Bronze.
• Apply ordered, idempotent merges in Silver.
• Track history where business requires it.
• Support late■arriving changes.

10. Scalability in Transformations

• Predicate pushdown and column pruning.
• Broadcast small dimensions.
• Pre■aggregate before joins where possible.
• Skew detection and mitigation strategies.

11. Reliability & Reprocessing

• Exactly■once business semantics via idempotent writes.
• Immutable raw data for replay.
• Targeted backfills instead of full recompute.
• Deterministic pipelines.

12. Benchmarking Strategy

• Throughput, latency, cost per TB, stability.
• Baseline, stress, and failure injection tests.
• Use benchmarks to drive partitioning and scaling decisions.

13. Serving Layer

• Gold tables optimized per consumer.
• BI: pre■aggregations and caching.
• ML: feature freshness and point■in■time correctness.
• APIs: low■latency access paths.

14. Monitoring & Observability

• Freshness, completeness, correctness SLOs.
• Lag, volume anomalies, quality drift.
• Cost monitoring and alerts.

15. Security & Governance

• RBAC, row/column■level security.
• PII tagging and masking.
• Lineage and auditability.

16. Best Practices Cheat Sheet

• Always land raw data immutably.
• Design pipelines to be re■runnable.
• Prefer fewer, larger files.
• Treat data quality as a first■class concern.
• Measure everything.

17. Interview Narrative Template

“I start by clarifying SLAs and data correctness requirements. I ingest data immutably into Bronze with contracts and quality checks, apply deterministic CDC/SCD logic in Silver with watermarking for late data, and publish consumer■optimized Gold tables. I ensure scalability via partitioning and parallelism, reliability via idempotent merges and replay, and operational excellence via benchmarking and SLO■driven monitoring.”