Q26. Design a data lake architecture.

I would design a data lake on object storage with raw, curated, and serving layers. Data is stored in open formats like Parquet, partitioned by date, with schema enforcement and cataloging for discoverability.

Q27. Design a data warehouse for analytics.

I would load curated data into a columnar warehouse optimized for analytical queries, define star schemas, and apply partitioning and clustering to reduce scan cost.

Q28. Design hot vs cold storage.

I would keep frequently queried data in hot storage and move older data to cheaper cold storage using lifecycle policies.

Q29. Design partitioning strategy for large tables.

I would partition by time and cluster by high-cardinality query keys to balance write and read efficiency.

Q30. Design schema evolution handling.

I would enforce backward-compatible changes using a schema registry and versioned schemas.

Q31. Design star vs snowflake schema.

I prefer star schema for simplicity and query performance, using snowflake only when dimensions are very large.

Q32. Design fact-dimension model.

I would store metrics in a fact table and descriptive attributes in dimensions to optimize analytical queries.

Q33. Design time-series data storage.

I would use time-based partitioning and a storage engine optimized for sequential writes and range queries.

Q34. Design multi-tenant data platform.

I would isolate tenants logically using namespaces, quotas, and access controls while sharing infrastructure.

Q35. Design SCD handling.

I would use SCD Type 2 for historical tracking with effective dates and current flags.

Q36. How would you scale a system to 10x traffic?

I would scale horizontally, add partitions, increase parallelism, and remove bottlenecks identified via metrics.

Q37. How do you handle data skew?

I would identify hot keys and mitigate using salting or repartitioning strategies.

Q38. How do you reduce shuffle cost in Spark?

I would partition data correctly, use narrow transformations, and avoid unnecessary joins.

Q39. Design low-latency query system.

I would pre-aggregate data, use indexes or clustering, and keep compute close to data.

Q40. Design cost-efficient analytics system.

I would separate compute and storage, use incremental processing, and apply retention policies.

Q41. Handle small files problem.

I would compact files periodically and control output file sizes.

Q42. Design high-cardinality aggregation.

I would pre-aggregate where possible and use approximate algorithms if acceptable.

Q43. Handle hot partitions.

I would rebalance partitions and distribute keys more evenly.

Q44. Design indexing strategy.

I would index frequently filtered columns while avoiding over-indexing.

Q45. Concurrent reads and writes handling.

I would rely on ACID-compliant table formats and snapshot isolation.

Q46. Design fault-tolerant pipeline.

I would add retries, checkpoints, and automatic recovery.

Q47. Handle partial failures.

I would isolate failures, retry idempotently, and alert operators.

Q48. Design retry and backoff.

I would use exponential backoff to avoid cascading failures.

Q49. Ensure idempotent writes.

I would use deterministic keys and upserts.

Q50. Ensure exactly-once delivery.

I would combine transactional writes with checkpointing.