**Databricks Spark Comprehensive Q&A Guide**

**🔹 Spark Core & Internals**

**Explain the lifecycle of a Spark job from user code → DAG → stages → tasks → executors.**

* User code → logical plan. Catalyst optimizer builds physical plan.
* Plan → DAG of stages (split at shuffle boundaries).
* Stage → multiple tasks (one per partition).
* Driver schedules tasks → Executors execute them → results back to driver.

**What are narrow vs wide transformations? Give practical examples.**

* Narrow: partition-local, no shuffle (map, filter).
* Wide: requires shuffle across partitions (groupByKey, join).

**How does Spark implement fault tolerance using RDD lineage?**

* Each RDD has lineage (a record of how it was derived).
* If partitions lost, Spark recomputes using lineage graph.
* Avoids full replication like Hadoop.

**What is the difference between RDD, DataFrame, and Dataset in Spark?**

* RDD: low-level, type-safe, no optimizations.
* DataFrame: high-level API, schema-aware, Catalyst optimizations.
* Dataset: type-safe (JVM languages), combines RDD + DataFrame features.

**How does the Catalyst optimizer work in Spark SQL? What are its main phases?**

* Analyzes SQL/DataFrame query.
* Phases: analysis (resolve attributes), logical optimization, physical planning, code generation.
* Improves performance with predicate pushdown, projection pruning.

**What are Spark accumulators and broadcast variables, and when would you use them?**

* Accumulators: write-only shared vars (counters, metrics).
* Broadcast: read-only cached data distributed to executors (e.g., lookup tables).

**Explain how lazy evaluation impacts performance in Spark.**

* Transformations are lazy, actions trigger execution.
* Allows Spark to optimize the DAG as a whole.
* Reduces redundant computations.

**🔹 Spark Execution & Optimization**

**What are the common shuffle operations in Spark? Why are they expensive?**

* Shuffle triggers: groupBy, join, reduceByKey, distinct.
* Expensive: disk IO + network transfer + serialization.

**How does Spark decide the number of shuffle partitions?**

* Default: 200 (spark.sql.shuffle.partitions).
* Can be overridden per job/config.
* AQE can dynamically coalesce.

**Explain how Adaptive Query Execution (AQE) improves Spark performance.**

* Dynamically adjusts plan at runtime.
* Merges small shuffle partitions, splits skewed ones.
* Chooses best join strategy (broadcast vs shuffle).

**What are the benefits and trade-offs of using Dynamic Allocation of executors in Databricks?**

* Pros: cost efficiency, auto-scale with workload.
* Cons: warm-up latency, possible executor churn.
* Works best with autoscaling clusters.

**How do you tune parallelism in Spark for optimal performance?**

* Match partitions ~ 2–3× total cores.
* Adjust spark.sql.shuffle.partitions.
* Use repartition()/coalesce() appropriately.

**What’s the difference between repartition() and coalesce()?**

* repartition(n): shuffle, can increase/decrease.
* coalesce(n): no shuffle, only decrease partitions.

**How do you identify and resolve data skew in Spark joins?**

* Symptoms: long-running stragglers in Spark UI.
* Fix: salting keys, broadcast small table, AQE skew handling.

**Explain speculative execution in Spark. How does it help with stragglers?**

* Detects slow tasks.
* Runs duplicate on another executor.
* First to finish is used, helps mitigate node slowness.

**🔹 Delta Lake (Core + Advanced)**

**What is the Delta transaction log and how does it guarantee ACID properties?**

* \_delta\_log/ stores JSON + checkpoints.
* Atomic commits with version numbers.
* Enables isolation, durability, consistency.

**How does schema evolution work in Delta Lake?**

* Auto add columns with mergeSchema.
* Manual: ALTER TABLE ADD COLUMNS.
* Strict enforcement prevents silent mismatches.

**What are MERGE INTO operations, and how do you handle SCD Type 1 vs Type 2 in Delta?**

* MERGE INTO supports UPSERT.
* Type 1: overwrite attributes.
* Type 2: expire old rows, insert new with effective\_from/to and is\_current.

**Explain Z-Ordering vs Partitioning. When would you use each?**

* Partitioning: coarse-grain (country, year).
* Z-Order: clustering within files (multi-col filters).
* Use both together for query pruning.

**What is vacuum in Delta Lake? Why do we use it, and what are the risks?**

* Deletes old versions/files → reduces storage.
* Default retention = 7 days.
* Risk: lose time travel before retention window.

**How do you use Time Travel in Delta?**

* Query older data by version or timestamp:

SELECT \* FROM tbl VERSION AS OF 5;

**What are Change Data Feed (CDF) features in Delta Lake?**

* table\_changes() returns row-level changes since version.
* Useful for incremental ETL, CDC pipelines.

**Compare Delta Lake vs traditional parquet tables.**

* Delta: ACID, schema enforcement, time travel, streaming.
* Parquet: just storage format, no transactions.

**🔹 Databricks Platform & Features**

**What are the differences between All-purpose clusters, Job clusters, and SQL warehouses?**

* All-purpose: interactive, notebooks, shared.
* Job clusters: ephemeral, tied to scheduled jobs.
* SQL warehouses: optimized for BI queries.

**How does Unity Catalog enforce row-level and column-level security?**

* Policies applied at catalog/schema/table/column level.
* Row filters and column masking.
* Central governance across workspaces.

**Explain how you’d organize a medallion architecture (Bronze, Silver, Gold) in Databricks.**

* Bronze: raw ingested data.
* Silver: cleaned + conformed.
* Gold: business aggregates.
* Each layer stored in Delta.

**What is the purpose of DBFS (Databricks File System)?**

* Abstraction layer for storage (S3/ADLS under the hood).
* Supports mounts for external buckets.
* Used for notebooks, libraries, checkpoints.

**How would you use Databricks Repos for CI/CD?**

* Connect to GitHub/Azure DevOps repos.
* Branching + PRs → promote notebooks to jobs.
* Integrates with pipelines.

**How do you configure Secrets Management in Databricks for secure API keys?**

* Store keys in secret scopes.
* Access with dbutils.secrets.get().
* Controlled via ACLs.

**Explain how Databricks Jobs differ from Workflows.**

* Jobs: define & schedule a task/notebook.
* Workflows: DAG of dependent tasks, orchestration.

**🔹 Streaming & Real-time**

**Difference between Structured Streaming and classic Spark Streaming.**

* Classic: DStream API, RDD micro-batches.
* Structured: DataFrame API, unified batch/stream, better guarantees.

**Explain how checkpointing works in Structured Streaming.**

* Stores offsets + state in checkpoint dir.
* Enables fault tolerance & exactly-once.

**How do you handle late-arriving data in Structured Streaming?**

* Use watermark with event-time windows.
* Discards old data beyond threshold.

**What are triggers in Structured Streaming (micro-batch, continuous)?**

* ProcessingTime (every N sec).
* Once (run once).
* AvailableNow (catch-up).
* Continuous (low-latency).

**How do you ensure exactly-once semantics in Structured Streaming with Delta sinks?**

* Delta sink + checkpointing → idempotent.
* Transactions guarantee no duplicates.

**What are the trade-offs between trigger.ProcessingTime and trigger.AvailableNow modes?**

* ProcessingTime: continuous latency.
* AvailableNow: batch catch-up then stop.
* Trade-off: latency vs throughput.

**🔹 Debugging & Troubleshooting**

**In Spark UI, how do you recognize a driver OOM vs executor OOM?**

* Driver OOM: whole job fails, driver logs show OutOfMemory.
* Executor OOM: specific tasks fail, retries happen.

**What does a Shuffle Fetch Failed error mean, and how do you debug it?**

* Lost/corrupted shuffle data.
* Causes: executor lost, disk full, skew.
* Debug: Spark UI → failed stage logs, executor tab.

**How do you debug job stuck in pending in Databricks?**

* Causes: no cluster resources, autoscaling delay.
* Fix: check cluster event logs, increase pool size.

**What’s your approach if you see disk spill (\_disk annotation) in Storage UI?**

* Means partitions spilled to disk.
* Fix: repartition data, increase memory, cache selectively.

**What is your debug playbook for a job with executor skew?**

* Identify skew in Spark UI (long tasks).
* Fix with salting, AQE skew splitting, broadcast joins.

**How do you debug straggler tasks?**

* Enable speculative execution.
* Investigate data locality, skew, executor health.

**What Spark configs would you tune for network retries and timeout issues?**

* spark.shuffle.io.maxRetries
* spark.shuffle.io.retryWait
* spark.network.timeout

**🔹 Design & Architecture (Hands-on Scenarios)**

**Design a data pipeline that ingests raw JSON from S3 → cleanses it → stores it as a Gold Delta table.**

* Bronze: ingest JSON → store raw Delta.
* Silver: apply schema, dedup, cleanse.
* Gold: aggregate + analytics-ready.

**How would you implement incremental ETL with Delta Lake (avoiding full reloads)?**

* Use MERGE INTO with CDC or watermark.
* Track high-water mark (e.g., timestamp).
* Upsert only changed rows.

**Describe how you’d model a fact + dimension star schema in Databricks.**

* Fact: transactional measures (fact\_orders).
* Dimensions: lookup tables (dim\_date, dim\_customer).
* Use surrogate keys + Delta.

**How do you handle slowly changing dimensions (SCD Type 2) in Delta tables?**

* Use MERGE INTO with is\_current, effective\_from/to.
* Expire old rows, insert new ones.

**If your data lake has billions of files, how do you optimize reads and writes?**

* Compact with OPTIMIZE.
* Use partitioning + Z-order.
* Enable auto compaction.

**What is your approach to designing a multi-tenant architecture in Databricks (governance + cost control)?**

* Separate catalogs/schemas in Unity Catalog.
* Tagging for cost tracking.
* Enforce row/column-level security.

**How would you integrate Databricks with external APIs in a Python notebook pipeline?**

* Use requests or httpx.
* Store keys in secret scopes.
* Write results into Delta.

**Explain how you’d design batch + streaming (Lambda or Kappa architecture) on Databricks.**

* Lambda: batch (ETL) + streaming (real-time).
* Kappa: unify into single streaming engine (Structured Streaming).
* Use Delta as sink for both.

**🔹 Advanced / Open-ended**

**If a Spark job takes 6 hours daily, what step-by-step optimizations would you try?**

* Cluster sizing → partitioning → skew fixes → caching → reduce shuffles → OPTIMIZE.

**How would you secure PII data in Databricks using Unity Catalog and Delta Lake?**

* Use column masking, row filters, role-based access.
* Encrypt data at rest/in motion.

**How would you explain to a stakeholder the difference between Data Lake vs Data Warehouse vs Lakehouse?**

* Lake: raw, flexible, cheap.
* WH: structured, governed, performant.
* Lakehouse: combines both with ACID + BI/ML.

**What’s your approach to cost optimization on Databricks clusters?**

* Use job clusters, autoscaling, spot instances.
* Tune partitions to avoid over-compute.
* Monitor with cost dashboards.

**You’re asked to support a machine learning workflow on Databricks. How would you prepare features and manage datasets?**

* Use Delta tables for feature storage.
* MLflow for experiment tracking.
* Databricks Feature Store for reuse.

**How do you integrate Databricks with Airflow or Camunda for workflow orchestration?**

* Airflow: Databricks provider → run jobs.
* Camunda: REST API tasks to trigger Databricks jobs.

**How would you design a high-availability setup for mission-critical pipelines in Databricks?**

* Multi-AZ clusters, retries, checkpointed streaming.
* Use pools for quick startup.
* Monitor with alerts.

**What are some anti-patterns in Databricks/Spark development you’ve seen, and how do you avoid them?**

* SELECT \* in production → prune columns.
* Over-caching → only cache reused data.
* Too many small files → compact with OPTIMIZE.
* Ignoring skew → leads to stragglers.