**12. Debugging & Production Issues**

**Scenario:**  
Your daily Spark job fails in production. Symptoms:

* Some tasks run fast, but a few are very slow (stragglers).
* Sometimes executors go OOM (out-of-memory).
* Analysts report missing partitions in Delta table.

How do you debug and fix?

**Baseline Approach (Expected in Interviews)**

1. **Check Spark UI / logs**

* Look at **stage DAG**: where is the bottleneck? (shuffle, join, skew).
* Review **task metrics**: shuffle read, spill, GC time.

1. **Common failure categories**

* **Skew**: one partition too large. Fix: salting, AQE.
* **OOM**: executor memory exhausted. Fix: repartition, filter early, increase memory.
* **Small files**: too many tiny files. Fix: OPTIMIZE/compaction.
* **Checkpoint corruption**: streaming jobs with invalid checkpoints. Fix: reset checkpoint carefully.

1. **Data issues**

* Missing partitions = ingestion failed or write incomplete.
* Quarantine bad data, reprocess partition.

**Advanced Considerations**

* **Adaptive Query Execution (AQE)**
  + Auto handles skew and repartitions at runtime.
  + Ensure spark.sql.adaptive.enabled = true.
* **Debugging joins**
  + Broadcast hints if small side fits in memory.
  + Repartition by join key to balance shuffle.
* **Delta-specific issues**
  + **Vacuum**: if retention too short, readers fail.
  + **Concurrent writes**: use Delta’s optimistic concurrency; avoid multiple jobs writing same table.
  + **Schema evolution issues**: if schema mismatches, enforce contract or evolve with mergeSchema.
* **Resiliency patterns**
  + Idempotent writes (MERGE).
  + Retry logic in orchestration.
  + Partial failures: process healthy partitions, quarantine bad ones.

**Follow-up Q&A**

**Q1. How do you debug a Spark OOM?**  
👉 Check if wide transformation (join/groupBy) caused large shuffle partition. Reduce partition size (repartition), filter earlier, increase executor memory, or use broadcast join.

**Q2. What causes straggler tasks?**  
👉 Data skew → one key has too many rows. Fix via salting, splitting heavy key, or enabling AQE.

**Q3. How do you debug missing partitions in a Delta table?**  
👉 Check if write job failed mid-way. Validate with \_delta\_log. Re-run for missing partitions. Use MERGE to avoid overwriting good data.

**Q4. How do you handle schema mismatch errors at runtime?**  
👉 Option 1: reject file (quarantine). Option 2: use mergeSchema with governance review. Always alert schema change.

**Q5. How do you recover from a corrupted checkpoint in streaming?**  
👉 If checkpoint is unrecoverable:

* Stop job, clear checkpoint, restart from known offset.
* Or rebuild state with compacted batch reprocessing.

**Q6. How do you prevent these issues proactively?**  
👉 Monitoring: Spark UI, Datadog/Prometheus, Delta audit tables.  
Set alerts on partition count, job duration, error rate.

**Cheat Sheet (Compressed Memory)**

* **Check Spark UI**: skew, OOM, shuffle, stragglers.
* **Skew:** salting, broadcast, AQE.
* **OOM:** repartition, filter early, executor memory.
* **Small files:** OPTIMIZE/compaction.
* **Delta issues:** schema drift, concurrency, vacuum.
* **Streaming issues:** checkpoint corruption → reset + replay.
* **Proactive:** monitoring, SLA alerts, quarantine bad data, retries.

✅ With Problem 12, your **full 12-problem library is complete**:

* Problems 1–6 → ingestion + processing + modeling.
* Problems 7–10 → quality + SQL perf + guardrails + orchestration robustness.
* Problem 11 → security & governance.
* Problem 12 → debugging & recovery.

You now have a **ready-to-use interview playbook**.