Coforge

DATA ENGINEER

Round 2 (Scenario-Based)

Experience range: 3–7 Years

1. Explain your end-to-end data pipeline in detail

Answer (What interviewers like):

In my project, data comes from multiple sources like relational databases and APIs. First, we ingest data into a **raw layer** in the Data Lake exactly as received, without transformations. This helps for audit and reprocessing.

Then we move data to a **processed/clean layer**, where we:

- Remove duplicates
- Handle nulls
- Apply business logic
- Standardize formats

Finally, we load curated data into a **Data Warehouse** where reporting and analytics teams consume it.

I also handle:

- Incremental logic
- Error handling
- Monitoring and alerts

This approach keeps the pipeline scalable and reliable.

2. Scenario: Business says pipeline is slow. How do you debug?

Detailed Answer:

First, I don't assume the issue. I **measure**.

Steps I follow:

1. Check which stage is slow (ingestion, transformation, or load)
2. Review logs and execution time
3. Validate data volume changes
4. Check for skewed joins or large shuffles
5. Analyze Spark UI or query plans

Once bottleneck is found, I optimize only that part instead of touching everything.

3. Scenario: How do you handle incremental loads in real projects?

Answer:

I always prefer incremental loads.

If the source provides:

- `last_updated_date` → I use watermark logic
- CDC → I use insert/update/delete flags

I store the **last successful run timestamp** and only process new or changed data.

For safety, I reprocess a small overlap window to capture late-arriving data.

4. Scenario: What happens if late data arrives after pipeline run?

Answer:

Late-arriving data is very common in production.

I solve it by:

- Reprocessing last 1–3 days of data
- Making pipelines idempotent so duplicates don't occur
- Tracking audit columns

This ensures accuracy without heavy reprocessing.

5. Scenario: Spark job failed with OutOfMemory error – what exactly do you do?

Answer:

I don't increase memory immediately.

First, I check:

- Are there wide transformations?

- Any skewed keys?
- Unnecessary caching?
- Large joins without filters?

Fixes include:

- Filtering early
- Repartitioning
- Using broadcast joins
- Reducing columns

Only if needed, I tune memory.

6. Scenario: Too many small files in Data Lake – why is it a problem?

Answer:

Too many small files increase metadata overhead and slow down Spark jobs.
Spark spends more time opening files than processing data.

Solution:

- Control file size during write
- Merge files using repartition/coalesce
- Run periodic compaction jobs

7. Scenario: Source sends duplicate records. How do you handle?

Answer:

First, I identify the **business key** (example: order_id).

Then:

- Deduplicate using window functions or Spark logic
- Keep latest record based on timestamp
- Log duplicates separately for audit

I never silently drop data.

8. Scenario: Schema changed in source and pipeline broke.

Answer:

First, I check whether change is:

- Additive (new column)
- Breaking (datatype change / column removed)

For additive changes:

- Enable schema evolution

For breaking changes:

- Version schema
- Communicate with downstream teams
- Backfill safely

9. Scenario: Report numbers don’t match source. How do you debug?

Answer:

I debug step-by-step:

1. Validate source count
2. Validate raw layer
3. Validate transformed layer
4. Validate warehouse tables
5. Check filters and joins

This systematic approach always reveals the issue.

10. Scenario: Job failed in middle of execution. What next?

Answer:

My pipelines are **idempotent**.

I:

- Use checkpoints
- Resume from last successful stage
- Avoid reprocessing everything

This saves time and prevents duplicates.

11. Scenario: How do you ensure data quality?

Answer:

I implement checks like:

- Null checks
- Range validation
- Duplicate detection
- Record count validation

Bad records go into a quarantine table, not into production.

12. Scenario: How do you monitor pipelines?

Answer:

I monitor:

- Job failures
- SLA breaches
- Sudden data spikes/drops

Alerts are configured so issues are detected early.

13. Scenario: How do you handle backfill for historical data?

Answer:

I never mix backfill with daily jobs.

I:

- Run backfill in controlled batches
- Monitor load
- Validate results before enabling daily pipeline

14. Scenario: How do you optimize SQL queries in warehouse?

Answer:

I:

- Avoid SELECT *
- Filter early
- Use proper joins
- Use window functions carefully
- Analyze execution plans

Performance tuning is data-driven, not guesswork.

15. Scenario: Data skew in Spark – how do you identify and fix?

Answer:

I check partition distribution.

If skew exists:

- I use salting

- Repartition data
- Optimize join keys

Balanced partitions = faster jobs.

16. Scenario: How do you handle PII data?

Answer:

I mask or encrypt sensitive fields.
Access is restricted using roles.
Only authorized users can see original data.

17. Scenario: Pipeline SLA missed repeatedly.

Answer:

I analyze historical runs to find patterns.
Then I optimize heavy steps or parallelize tasks.
If needed, I redesign the pipeline.

18. Scenario: How do you deploy changes safely?

Answer:

- Version control with Git
- Dev → Test → Prod promotion
- Rollback plan always ready

19. Scenario: Interviewer asks —"What makes you a good Data Engineer?"

Answer:

I focus on **reliability, performance, and clarity**.
I take ownership of pipelines and fix issues proactively.
I also communicate clearly with business teams.

20. Final Question: Why Coforge should hire you?

Answer:

I understand real-world data problems and solve them practically.
I build scalable pipelines, handle failures confidently, and continuously optimize systems.