

# DATA PIPELINE FAILURES

## Real Interview Scenarios & How to Handle Them

---

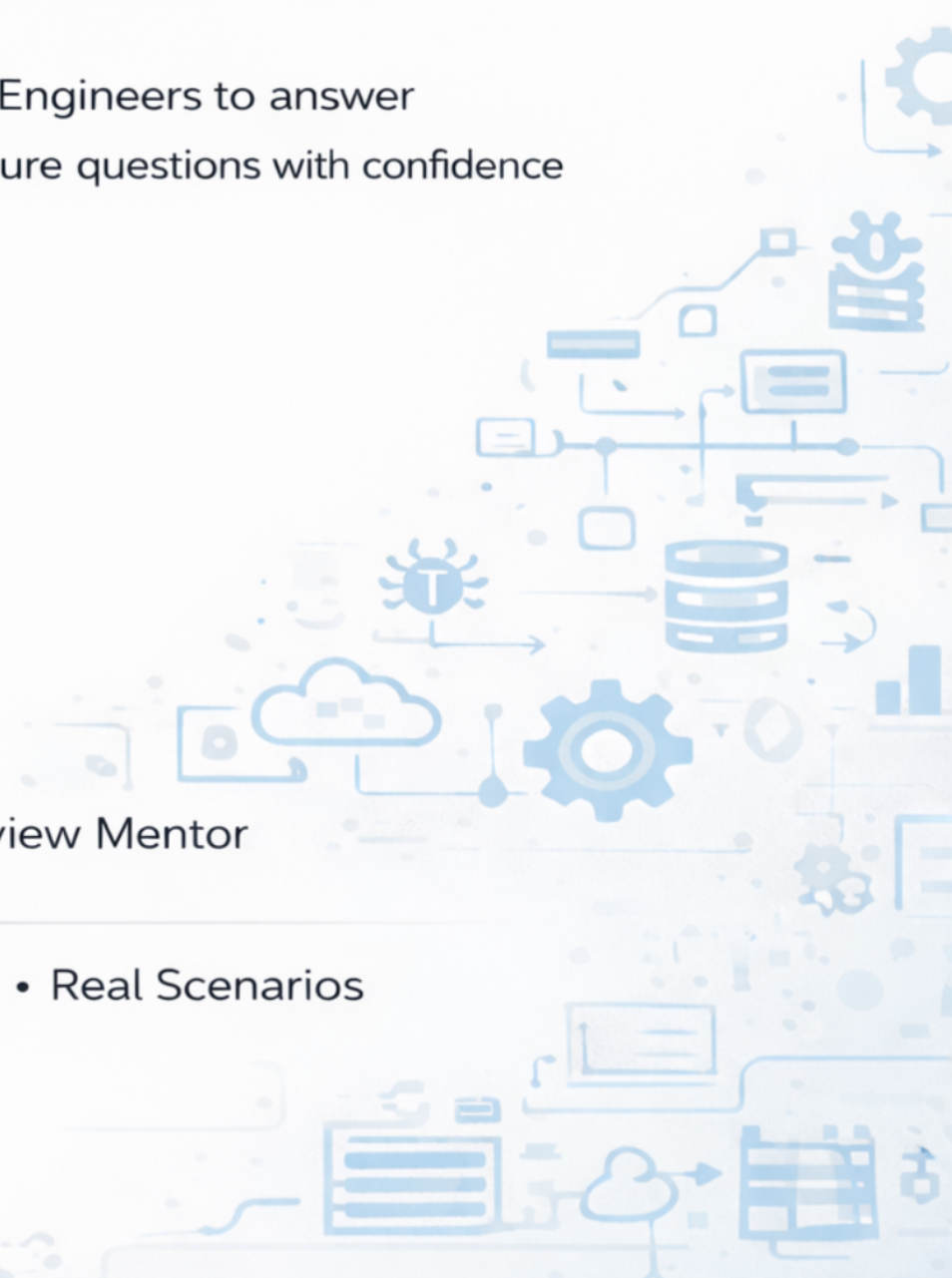
A practical guide for Data Engineers to answer  
real-world data pipeline failure questions with confidence

**By Ankita Gulati**

Senior Data Engineer | Interview Mentor

---

Interview Edition • Practical • Real Scenarios



# Table Of Content

<b>Scenario 1.....</b>	<b>5</b>
<b>Daily Spark Runtime Explodes.....</b>	<b>5</b>
Problem Statement.....	5
Clarifying Questions.....	5
Clarifying Information & Assumptions.....	5
Key Observation.....	6
Solution: Step-by-Step Deep Explanation.....	6
Final Resolution.....	8
Key Learning Outcomes.....	8
Core Principle Reinforced.....	8
<b>Scenario 2.....</b>	<b>9</b>
<b>Intermittent Spark Job Failures.....</b>	<b>9</b>
Problem Statement.....	9
Clarifying Questions.....	9
Clarifying Information & Assumptions.....	9
Key Observation.....	10
Investigation & Root Cause Analysis.....	10
Final Resolution.....	12
Key Learnings.....	12
Core Principle Reinforced.....	12
<b>Scenario 3.....</b>	<b>13</b>
<b>Upstream Dependency Failure Blocking the DAG.....</b>	<b>13</b>
Problem Statement.....	13
Clarifying Questions.....	13
Clarifying Information & Assumptions.....	13
Key Observation.....	14
Investigation & Root Cause Analysis.....	14
Final Resolution.....	16
Key Learnings.....	16
Core Principle Reinforced.....	16

<b>Scenario 4.....</b>	<b>17</b>
<b>Partial Job Success Causing Silent Data Gaps.....</b>	<b>17</b>
Problem Statement.....	17
Clarifying Questions.....	17
Confirmed Facts & Assumptions.....	17
Key Observation.....	18
Root Cause Analysis.....	18
Final Resolution.....	20
Key Learnings.....	20
Core Principle Reinforced.....	20
<b>Scenario 5.....</b>	<b>21</b>
<b>Hidden Upstream Schema Change Breaking a Production Pipeline.....</b>	<b>21</b>
Problem Statement.....	21
Clarifying Questions.....	21
Confirmed Facts & Assumptions.....	21
Key Observation.....	22
Root Cause Analysis.....	22
Final Resolution.....	24
Key Learnings.....	24
Core Principle Reinforced.....	24
<b>Scenario 6.....</b>	<b>25</b>
<b>Batch Job Fails Only on Large Input Files.....</b>	<b>25</b>
Problem Statement.....	25
Clarifying Questions.....	25
Confirmed Facts & Assumptions.....	25
Key Observation.....	26
Root Cause Analysis.....	26
Final Resolution.....	28
Key Learnings.....	28
Core Principle Reinforced.....	28
<b>Scenario 7.....</b>	<b>29</b>
<b>Upstream Data Delay Breaking Downstream SLA.....</b>	<b>29</b>
Problem Statement.....	29
Clarifying Questions.....	29
Confirmed Facts & Assumptions.....	29
Key Observation.....	30
Root Cause Analysis.....	30
Final Resolution.....	32
Key Learnings.....	32
Core Principle Reinforced.....	32

## Scenario 1

# Daily Spark Runtime Explodes

### Problem Statement

You are responsible for a daily Spark batch job that processes approximately 1 TB of data.

- Normal runtime: 20 minutes
- Current runtime: 90 minutes
- Downstream business reports are blocked

The job does not fail but runs significantly slower than usual

**Objective:** Diagnose the root cause and bring the job back within the 30-minute SLA.

### Clarifying Questions

Before proposing solutions, the engineer must reduce ambiguity.

Key questions include:

- Did the data volume or distribution change?
- Were there schema, join, or aggregation changes?
- Are all Spark stages slow or only specific ones?
- Do some tasks take much longer than others?
- Are executors underutilized or stuck waiting?

### Clarifying Information & Assumptions

After initial investigation, the following facts are confirmed:

- Data volume increased by ~15%
- No changes in code, schema, or Spark version
- Cluster configuration remains unchanged
- No job failures or retries occurred

These assumptions indicate the slowdown is **not expected growth-related behavior**.

## Key Observation

A **15% increase in data volume** should result in a small, near-linear increase in runtime.

A 4–5× runtime increase strongly suggests:

- Uneven data distribution
- Execution imbalance across Spark tasks
- Presence of straggler tasks

This shifts focus from scaling resources to execution behavior analysis.

## Solution: Step-by-Step Deep Explanation

### Step 1: Analyze Job Execution Using Spark UI

The first step is to inspect how Spark is executing the job, not how much data it processes. Key indicators:

- Large gap between average task time and maximum task time
- One or two tasks running significantly longer than others
- Several executors idle while a few remain busy

This pattern confirms that a small subset of tasks is dominating total runtime.

### Step 2: Understand Why Straggler Tasks Occur

Spark divides data into **partitions**, and each task processes one partition.

Stragglers occur when:

- Some partitions contain **disproportionately large data**
- Those partitions take much longer to process
- The entire waits for the slowest task to finish

This behavior is a classic symptom of **data skew**.

### Step 3: What Data Skew Means Conceptually

Data skew happens when:

- A small number of keys appear extremely frequently
- Joins or aggregations group large volumes of data under those keys
- Spark assigns those heavy keys to one or few