

Interview Roadmap

What a data engineering interview loop tests, which handbook guides cover each part, and a four-week plan to prepare.

Last reviewed 29 Sep 2026 ·

Related: [SQL Interview Patterns](https://sarangambekar1997.github.io/data-engineering-handbook/09-interviews/sql-interview-patterns/) (https://sarangambekar1997.github.io/data-engineering-handbook/09-interviews/sql-interview-patterns/) · [System Design Case Studies](https://sarangambekar1997.github.io/data-engineering-handbook/09-interviews/system-design-case-studies/) (https://sarangambekar1997.github.io/data-engineering-handbook/09-interviews/system-design-case-studies/) · [System Design](https://sarangambekar1997.github.io/data-engineering-handbook/08-architecture/system-design/) (https://sarangambekar1997.github.io/data-engineering-handbook/08-architecture/system-design/) · [Glossary](https://sarangambekar1997.github.io/data-engineering-handbook/99-reference/glossary/) (https://sarangambekar1997.github.io/data-engineering-handbook/99-reference/glossary/)

Overview

Challenge: Data engineering interviews cover a wide range: SQL, Python, data modelling, pipelines, distributed systems, cloud, and increasingly LLM tooling. Preparing "everything" is impossible, and reading without practising does not transfer to a live round.

Solution: Know the format of the loop, map each round to the topics that actually get asked, and practise the *skill* each round tests. Every guide in this handbook ends with Interview Questions, so this page tells you which ones to prioritise and in what order.



Loops differ by company: some skip a round, add a take-home or a live debugging session, or add an AI/LLM round. Ask the recruiter for the format and the interview length.

On this page

Basic - [The Rounds](https://sarangambekar1997.github.io/data-engineering-handbook/09-interviews/interview-roadmap/#the-rounds) (https://sarangambekar1997.github.io/data-engineering-handbook/09-interviews/interview-roadmap/#the-rounds) - [Topic Map](https://sarangambekar1997.github.io/data-engineering-handbook/09-interviews/interview-roadmap/#topic-map) (https://sarangambekar1997.github.io/data-engineering-handbook/09-interviews/interview-roadmap/#topic-map)

Intermediate - [A Four-Week Plan](https://sarangambekar1997.github.io/data-engineering-handbook/09-interviews/interview-roadmap/#a-four-week-plan) (https://sarangambekar1997.github.io/data-engineering-handbook/09-interviews/interview-roadmap/#a-four-week-plan) - [How to Answer](https://sarangambekar1997.github.io/data-engineering-handbook/09-interviews/interview-roadmap/#how-to-answer) (https://sarangambekar1997.github.io/data-engineering-handbook/09-interviews/interview-roadmap/#how-to-answer)

Advanced - [Behavioural Round](https://sarangambekar1997.github.io/data-engineering-handbook/09-interviews/interview-roadmap/#behavioural-round) (https://sarangambekar1997.github.io/data-engineering-handbook/09-interviews/interview-roadmap/#behavioural-round) - [Take-Homes and Live Debugging](https://sarangambekar1997.github.io/data-engineering-handbook/09-interviews/interview-roadmap/#take-homes-and-live-debugging) (https://sarangambekar1997.github.io/data-engineering-handbook/09-interviews/interview-roadmap/#take-homes-and-live-debugging) - [Questions to Ask Them](https://sarangambekar1997.github.io/data-engineering-handbook/09-interviews/interview-roadmap/#questions-to-ask-them) (https://sarangambekar1997.github.io/data-engineering-handbook/09-interviews/interview-roadmap/#questions-to-ask-them)

Reference - [Common Pitfalls](https://sarangambekar1997.github.io/data-engineering-handbook/09-interviews/interview-roadmap/#common-pitfalls) (https://sarangambekar1997.github.io/data-engineering-handbook/09-interviews/interview-roadmap/#common-pitfalls) - [Cheat Sheet](https://sarangambekar1997.github.io/data-engineering-handbook/09-interviews/interview-roadmap/#cheat-sheet) (https://sarangambekar1997.github.io/data-engineering-handbook/09-interviews/interview-roadmap/#cheat-sheet) - [Interview Questions](https://sarangambekar1997.github.io/data-engineering-handbook/09-interviews/interview-roadmap/#interview-questions) (https://sarangambekar1997.github.io/data-engineering-handbook/09-interviews/interview-roadmap/#interview-questions) - [Further Reading](https://sarangambekar1997.github.io/data-engineering-handbook/09-interviews/interview-roadmap/#further-reading) (https://sarangambekar1997.github.io/data-engineering-handbook/09-interviews/interview-roadmap/#further-reading)

The Rounds

Round	What it tests	Typical content	Prepare with
Recruiter / hiring manager	Fit, level, communication	Your projects, why this role, scale you have handled	A two-minute story for each project
SQL	Fluency and pattern recognition	Window functions, joins, dedup, cohorts, sessionisation	SQL Interview Patterns (https://sarangambekar1997.github.io/data-engineering-handbook/09-interviews/sql-interview-patterns/), Lab 01 (https://github.com/sarangambekar1997/data-engineering-handbook/tree/main/labs/01-sql-analytics)
Python / coding	Data wrangling and clean code	Parse and aggregate records, dictionaries and sets, generators, sometimes a small algorithm	Python for DE (https://sarangambekar1997.github.io/data-engineering-handbook/00-foundations/python-reference/)
Data modelling	Turning a business into tables	Star schema, slowly changing dimensions, grain, keys	Data Modeling (https://sarangambekar1997.github.io/data-engineering-handbook/01-storage/data-modeling/)

Round	What it tests	Typical content	Prepare with
Pipeline design	Reliable ETL	Idempotency, incremental loads, backfills, late data, quality checks	DE Concepts (https://sarangambekar1997.github.io/data-engineering-handbook/00-foundations/de-concepts/), Ingestion & CDC (https://sarangambekar1997.github.io/data-engineering-handbook/02-processing/ingestion-cdc/), Airflow (https://sarangambekar1997.github.io/data-engineering-handbook/03-orchestration/airflow-reference/), Testing and CI/CD (https://sarangambekar1997.github.io/data-engineering-handbook/06-infrastructure/testing-cicd/)
System design	Architecture and trade-offs	Batch vs streaming, storage choices, scale, cost	System Design (https://sarangambekar1997.github.io/data-engineering-handbook/08-architecture/system-design/), Case Studies (https://sarangambekar1997.github.io/data-engineering-handbook/09-interviews/system-design-case-studies/)
Tool depth	Real experience with the stack on your CV	Spark tuning, dbt, Kafka semantics, warehouse internals	The guide for each tool
Behavioural	Ownership and judgement	An incident you handled, a disagreement, a mistake	Behavioural Round (https://sarangambekar1997.github.io/data-engineering-handbook/09-interviews/interview-roadmap/#behavioural-round)

Topic Map

Where the questions come from, ordered by how often they appear in practice.

Priority	Topic	Guides
Almost always	SQL and window functions	SQL (https://sarangambekar1997.github.io/data-engineering-handbook/00-foundations/sql-reference/), SQL Interview Patterns (https://sarangambekar1997.github.io/data-engineering-handbook/09-interviews/sql-interview-patterns/)
	Data modelling (star schema, SCD)	Data Modeling (https://sarangambekar1997.github.io/data-engineering-handbook/01-storage/data-modeling/)

Priority	Topic	Guides
	ETL vs ELT, batch vs streaming, idempotency	DE Concepts (https://sarangambekar1997.github.io/data-engineering-handbook/00-foundations/de-concepts/)
	Python data handling	Python for DE (https://sarangambekar1997.github.io/data-engineering-handbook/00-foundations/python-reference/)
Very often	Spark internals: shuffles, skew, partitioning	PySpark (https://sarangambekar1997.github.io/data-engineering-handbook/02-processing/pyspark-reference/)
	Orchestration and backfills	Airflow (https://sarangambekar1997.github.io/data-engineering-handbook/03-orchestration/airflow-reference/), Dagster (https://sarangambekar1997.github.io/data-engineering-handbook/03-orchestration/dagster-reference/)
	Warehouse and lakehouse choices	Snowflake (https://sarangambekar1997.github.io/data-engineering-handbook/01-storage/snowflake-reference/), BigQuery (https://sarangambekar1997.github.io/data-engineering-handbook/01-storage/bigquery-reference/), Delta Lake (https://sarangambekar1997.github.io/data-engineering-handbook/01-storage/delta-lake/), Iceberg (https://sarangambekar1997.github.io/data-engineering-handbook/01-storage/apache-iceberg/)
	Data quality and observability	Data Quality (https://sarangambekar1997.github.io/data-engineering-handbook/05-quality-governance/data-quality/), Pipeline Observability (https://sarangambekar1997.github.io/data-engineering-handbook/05-quality-governance/pipeline-observability/)
Often	Kafka and streaming semantics	Kafka (https://sarangambekar1997.github.io/data-engineering-handbook/04-streaming/kafka-reference/), Flink (https://sarangambekar1997.github.io/data-engineering-handbook/04-streaming/flink-reference/)
	dbt	dbt (https://sarangambekar1997.github.io/data-engineering-handbook/02-processing/dbt-reference/)
	Testing and CI/CD for pipelines	Testing and CI/CD (https://sarangambekar1997.github.io/data-engineering-handbook/06-infrastructure/testing-cicd/)
	Cost and performance	Cost Optimization (https://sarangambekar1997.github.io/data-engineering-handbook/08-architecture/cost-optimization/)
	Serving fast analytics, query federation	Real-Time Analytics Databases (https://sarangambekar1997.github.io/data-engineering-handbook/01-storage/realtime-olap/), Trino (https://sarangambekar1997.github.io/data-engineering-handbook/02-processing/trino-federation/)

Priority	Topic	Guides
	Running data workloads on Kubernetes	Kubernetes (https://sarangambekar1997.github.io/data-engineering-handbook/06-infrastructure/kubernetes-for-de/)
	Event time, windows and late data	Beam and Dataflow (https://sarangambekar1997.github.io/data-engineering-handbook/04-streaming/beam-dataflow/), Streaming SQL (https://sarangambekar1997.github.io/data-engineering-handbook/04-streaming/streaming-sql/)
	NoSQL modelling and <u>CDC</u> from operational stores	NoSQL and Operational Stores (https://sarangambekar1997.github.io/data-engineering-handbook/01-storage/nosql-operational-stores/)
	Serving data to the business	BI Tools (https://sarangambekar1997.github.io/data-engineering-handbook/02-processing/bi-tools/)
	Incidents, on-call and postmortems	DataOps (https://sarangambekar1997.github.io/data-engineering-handbook/05-quality-governance/dataops-operations/)
	Catalogs, ownership and lineage tooling	Data Catalogs in Practice (https://sarangambekar1997.github.io/data-engineering-handbook/05-quality-governance/data-catalogs/)
	Choosing and defending a stack	Choosing a Stack (https://sarangambekar1997.github.io/data-engineering-handbook/08-architecture/choosing-a-stack/)
	LLMs and data: safe SQL generation, <u>MCP</u>	MCP and Text-to-SQL (https://sarangambekar1997.github.io/data-engineering-handbook/07-ai/mcp-text-to-sql/)
	Cloud platform specifics (Microsoft)	Azure and Fabric (https://sarangambekar1997.github.io/data-engineering-handbook/01-storage/azure-fabric/)
	Security and privacy	Data Security & Privacy (https://sarangambekar1997.github.io/data-engineering-handbook/05-quality-governance/data-security-privacy/)
Growing	LLMs, RAG, evals	LLM APIs (https://sarangambekar1997.github.io/data-engineering-handbook/07-ai/llm-apis/), RAG (https://sarangambekar1997.github.io/data-engineering-handbook/07-ai/rag/), Evals (https://sarangambekar1997.github.io/data-engineering-handbook/07-ai/eval-and-evals/)

Priority	Topic	Guides
Situational	Terraform, Docker, CDC, lineage	Terraform (https://sarangambekar1997.github.io/data-engineering-handbook/06-infrastructure/terraform-for-de/), Docker (https://sarangambekar1997.github.io/data-engineering-handbook/06-infrastructure/docker-reference/), Governance (https://sarangambekar1997.github.io/data-engineering-handbook/05-quality-governance/governance-lineage/)

Go deep on the tools listed on your CV first. Interviewers ask about what you claim.

A Four-Week Plan

About 6–8 hours a week. Adjust the pace to your interview date.

Week	Focus	Do
1: Core skills	SQL and Python	Type out all 15 SQL patterns (https://sarangambekar1997.github.io/data-engineering-handbook/09-interviews/sql-interview-patterns/) from memory. Solve five problems a day for a week. Do Lab 01 (https://github.com/sarangambekar1997/data-engineering-handbook/tree/main/labs/01-sql-analytics). Refresh Python dictionaries, comprehensions, generators and <code>collections</code>
2: Modelling and pipelines	Data modelling, ETL design	Model an e-commerce and a ride-sharing business on paper, with grain, facts and dimensions. Explain idempotent loads, backfills and late-arriving data out loud. Do Lab 02 (https://github.com/sarangambekar1997/data-engineering-handbook/tree/main/labs/02-dbt-transformations) and Lab 05 (https://github.com/sarangambekar1997/data-engineering-handbook/tree/main/labs/05-airflow-orchestration)
3: Distributed systems	Spark, streaming, storage	Read the Spark shuffle, skew and join sections and explain them without notes. Kafka delivery guarantees and consumer groups. Lakehouse table formats. Do Lab 03 (https://github.com/sarangambekar1997/data-engineering-handbook/tree/main/labs/03-spark-lakehouse) and Lab 04 (https://github.com/sarangambekar1997/data-engineering-handbook/tree/main/labs/04-kafka-streaming)
4: Design and stories	System design, behavioural	Work through each case study (https://sarangambekar1997.github.io/data-engineering-handbook/09-interviews/system-design-case-studies/) aloud, timed at 45 minutes. Write five STAR stories. Do two full mock interviews with a friend or recorded

The most valuable habit: **speak your reasoning out loud** while you practise. Silent practice does not train the skill that the interview measures.

How to Answer

A repeatable structure works for almost any technical question:

1. **Clarify.** Ask about data volume, freshness, correctness needs, `NULL` s and duplicates, and who consumes the result.
2. **State assumptions.** "I'll assume orders are unique by `order_id` and time is UTC."
3. **Outline the approach** in one or two sentences before you write anything.
4. **Solve simply first**, then refine. A correct simple answer beats an unfinished clever one.
5. **Test with a tiny example**, including edge cases: empty input, ties, `NULL` s, one row.
6. **Discuss trade-offs and scale:** what changes with 1000× the data? What would you monitor?

For system design, use the sequence in [System Design](https://sarangambekar1997.github.io/data-engineering-handbook/08-architecture/system-design/) (<https://sarangambekar1997.github.io/data-engineering-handbook/08-architecture/system-design/>): requirements, capacity, architecture, deep dive on the risky parts, failure modes, cost. Draw as you talk.

If you get stuck, say what you are thinking. Interviewers give hints to candidates who communicate, and rarely to those who go silent.

Behavioural Round

Prepare **five stories** you can adapt to most questions. For each, use STAR: **S**ituation, **T**ask, **A**ction, **R**esult. Spend most of the time on *your* actions and a measurable result.

Story	Typical question it answers
A pipeline failure or data incident you handled	"Tell me about a time something broke in production"
A pipeline or query you made much faster or cheaper	"Tell me about an optimisation"
A disagreement about a technical decision	"How do you handle conflict?"
A project with unclear requirements	"How do you work with stakeholders?"
A mistake you made and what you changed after	"Tell me about a failure"

Good signals: you took ownership, you communicated early, you fixed the root cause and added a monitor or test, and you can name a number (hours saved, cost reduced, latency cut). Read [Pipeline Observability](https://sarangambekar1997.github.io/data-engineering-handbook/05-quality-governance/pipeline-observability/) (https://sarangambekar1997.github.io/data-engineering-handbook/05-quality-governance/pipeline-observability/) for the incident vocabulary.

Take-Homes and Live Debugging

Take-homes are judged on engineering habits more than cleverness: - Make it runnable in one command, with a README that states assumptions and how to run it. - Idempotent loads, sensible schema, tests for the tricky logic, and data quality checks. - Keep it small and finished. Mention what you would do with more time (monitoring, incremental loads, scaling).

Live debugging gives you a broken query or pipeline. Read the error, form a hypothesis, check the smallest thing that confirms it, and narrate as you go. Common causes: fan-out joins, `NULL` handling, wrong grain, a timezone shift, a schema change, duplicates from a retry.

Questions to Ask Them

Good questions show judgement and help you decide whether to join.

- What does the data stack look like, and what is painful about it today?
 - How do you find out when a pipeline is wrong, and how long does it take?
 - What is the on-call arrangement for data incidents?
 - How are schema changes in source systems communicated to the data team?
 - How do data engineers work with analysts, scientists and product?
 - What would success look like for this role in six months?
-

Common Pitfalls

Pitfall	Symptom	Fix
Memorising tool trivia	You can recite settings but not explain <i>why</i>	Learn the problem each tool solves and its trade-offs

Pitfall	Symptom	Fix
Starting to code immediately	Solving the wrong problem	Clarify and state assumptions first
Ignoring <code>NULL</code> s, duplicates, ties and time zones	Correct on the happy path only	Ask about them, then handle them
Silent thinking	Interviewer cannot help or assess you	Narrate your reasoning
One-tool answers to design questions	"Just use Spark" for everything	Compare at least two options with trade-offs
No numbers in stories	Impact sounds vague	Quantify: rows, hours, dollars, latency
Claiming tools you have only read about	An easy follow-up exposes it	List what you have used, and be honest about depth

Cheat Sheet

Situation	Do this
Question is vague	Ask about scale, freshness, consumers, correctness
SQL problem	Name the pattern, write the simple version, test on a tiny table
Pipeline design	Sources → ingest → store → transform → serve, then idempotency, backfill, quality, monitoring
"How would it scale?"	Where is the bottleneck: shuffle, skew, single writer, hot partition?
"What could go wrong?"	Late data, duplicates, schema change, retries, source outage
Out of time	Summarise the design and list what you would do next
Stuck	Say what you tried, and ask a clarifying question

Interview Questions

Q: Walk me through a pipeline you built end to end. A: Give the context and scale, then follow the data: source, ingestion (batch or `CDC`), storage layers, transformations, orchestration, quality checks, monitoring, consumers. Highlight one hard problem you solved (late data, a slow join, cost) with a number for the result, and say what you would change now.

Q: How do you make a pipeline idempotent? A: Design each run so that running it twice for the same input gives the same result: overwrite a partition or `MERGE` on a business key instead of appending blindly, avoid side effects that cannot be repeated, and use deterministic keys. This makes retries and backfills safe.

Q: What would you check first when a dashboard number looks wrong? A: Whether the data is fresh and complete (freshness and row counts by layer), then whether the logic changed (recent deploys, schema changes), then grain and join issues such as fan-out or duplicates. Lineage tells me where to start, and I compare against a trusted source to confirm the fix.

Q: How do you decide between batch and streaming? A: By the freshness the business needs and the cost of being wrong or late. If hourly or daily is enough, batch is simpler and cheaper. Streaming is worth it for use cases with minutes or seconds of tolerance, such as fraud detection or operational alerts, and it adds state, ordering and exactly-once questions.

Further Reading

- [System Design](https://sarangambekar1997.github.io/data-engineering-handbook/08-architecture/system-design/) (https://sarangambekar1997.github.io/data-engineering-handbook/08-architecture/system-design/)
- *Designing Data-Intensive Applications* — Martin Kleppmann (O'Reilly)
- *Fundamentals of Data Engineering* — Joe Reis & Matt Housley (O'Reilly)
- *The Data Warehouse Toolkit* — Ralph Kimball & Margy Ross (Wiley)
- [Data Engineering Zoomcamp](https://github.com/DataTalksClub/data-engineering-zoomcamp) (https://github.com/DataTalksClub/data-engineering-zoomcamp): free, project-based course

Previous: [Cost Optimization](https://sarangambekar1997.github.io/data-engineering-handbook/08-architecture/cost-optimization/) (https://sarangambekar1997.github.io/data-engineering-handbook/08-architecture/cost-optimization/) • **Next:** [SQL Interview Patterns](https://sarangambekar1997.github.io/data-engineering-handbook/09-interviews/sql-interview-patterns/) (https://sarangambekar1997.github.io/data-engineering-handbook/09-interviews/sql-interview-patterns/) • **Back to:** [Index](https://sarangambekar1997.github.io/data-engineering-handbook/) (https://sarangambekar1997.github.io/data-engineering-handbook/)

 September 29, 2026