make my trip

Data Engineering Interview Questions

Ankita Gulati

Shubh Goyal

Job Details

- **Position:** Data Engineer II
- **Experience:** 3+ years
- **Location:** Bangalore
- **Work mode:** Office
- **Compensation:** ₹25+ LPA
- **Total Rounds:** 3
- **Top Required Skills:**

1. SQL
2. PySpark / Python
3. Cloud Data Engineering
4. ETL / Data Modeling
5. Big Data & Streaming
6. System Design

Ankita Gulati

Shubh Goyal

Round 1
Core CS & Data Structures

1. What is the most frequently used data structure in your projects? (Candidate example: HashMap).
2. Explain how a HashMap works internally.
3. How would you design a good key for a HashMap?
4. Compare HashMap vs. Hashtable — what are the key differences?
5. How are collisions handled in HashMap?
6. What is the time complexity of the get() operation in HashMap under average and worst-case scenarios?
7. Write code to check whether a given string of brackets is balanced or not.
   - → Example Input: {[]} → balanced
   - → Example Input: {[()]} → not balanced
8. Explain differences and use cases for HashMap, ConcurrentHashMap, and LinkedList.
9. Be prepared to illustrate concepts on the whiteboard with diagrams and sample code.

Ankita Gulati　　　　Shubh Goyal

Round 2
Coding & Problem-Solving

Problem 1 — LRU Cache

1. Implement an LRU (Least Recently Used) cache in Java (or Python).
2. What data structures would you use? Justify your choice.
   - Why HashMap?
   - Why Doubly Linked List?
3. Draw the design on the whiteboard and explain how get() and put() operations would work.
4. Write pseudo-code before the actual implementation, then complete the code.
5. Discuss time and space complexities of your solution.

Ankita Gulati Shubh Goyal

Problem 2 — Random Secure Password Generator

1. Write a program to generate a random secure password given constraints:
   - Minimum and maximum length
   - Required number of uppercase and lowercase letters
   - Required number of special characters and digits
2. Justify your approach and show how you would test edge cases (e.g., when constraints cannot be satisfied).

Round 3
Big Data & Distributed Systems

1. Explain the life cycle of a MapReduce job. Describe each phase in detail.
2. Write code (pseudo or real) to process CDR (Call Detail Record) data using MapReduce.
3. What is speculative execution in MapReduce? Why is it needed?
4. How do you run Spark on YARN? Compare Cluster Mode vs. Client Mode.
5. Explain fault tolerance in RDDs — how does Spark handle node failures?
6. Differentiate between Actions and Transformations in Spark, with examples.
7. Compare map() vs. flatMap() with a use-case example.
8. What is a broadcast variable in Spark? When would you use it?

Ankita Gulati Shubh Goyal

9. What are accumulators in Spark? How are they different from broadcast variables?
10. Explain the data flow of a Spark job execution.
11. What is a DAG (Directed Acyclic Graph) in Spark, and how is it formed?
12. Compare cache() vs. persist() in Spark. When would you use each?
13. Give some real-world use cases for Spark Streaming.

Ankita Gulati Shubh Goyal

Round 4
Foundational Statistics

1. Given a wide table with missing values, write a Python (pandas) function to impute_missing by column using a combination of median for numeric columns and mode for categoricals. Explain OOM mitigation strategies for large tables.
2. Implement feature engineering steps in Python for a time-series ML model: lag features, rolling means, and time-since-last-event. Show vectorized solution and explain why it's preferred over apply.
3. Data reshaping: show how to pivot and melt dataframes for a model input and explain memory-efficient approaches for very large data (e.g., chunking, dtypes downcasting).
4. Write a generator-based Python function that streams CSV rows, applies lightweight transformations, and yields compressed Parquet file parts for downstream processing.

Ankita Gulati　　　　Shubh Goyal

5. Explain how you would find and fix a hotspot in Python ETL that spends most time in deserialization. Mention tools (profiler, memory profiler) and strategies (C extensions, multiprocessing, PySpark).
6. When and how would you use multithreading vs. multiprocessing vs. async IO in an ETL task? Give a code sketch for each case.
7. Unit/integration testing: write a pytest unit test for a transformation function and describe how you'd mock external data sources.
8. Data validation: design a function that asserts data contracts (column presence, data types, row-count thresholds) and integrates with CI/CD.
9. Advanced Python coding prompt (live): given a log file feed, implement an efficient deduplication + last-seen aggregator that uses bounded memory (explain algorithm/design).
10. Explain how you'd implement idempotent transformations in your Python ETL jobs.

Round 5
Data Structures & Algorithms

1. What are the differences between RDD, DataFrame, and Dataset in Spark? When would you use each?
2. Explain Catalyst optimizer and Tungsten — how do they help performance?
3. repartition() vs coalesce() — describe semantics and give concrete use cases.
4. Explain broadcast joins vs shuffle joins and when to use each; include cost trade-offs.
5. How do you detect and fix data skew in a Spark job? Provide at least three mitigation strategies.
6. Explain checkpointing and write-ahead logs — when are they required in streaming jobs?
7. Spark streaming: explain event-time processing, watermarks, and late-arriving data handling. Provide a code sketch for windowed aggregations with allowed lateness.
8. How do you design Spark jobs to be fault-tolerant and to provide exactly-once semantics (if possible)? Discuss idempotency and external sinks.

Ankita Gulati Shubh Goyal

9. Partitioning strategy: how would you choose partition keys for a fact table intended for daily batch processing and frequent user-level queries?

10. Implement a Spark job that merges an hourly batch into a daily partitioned table while ensuring schema evolution and manifest updates.

Round 6
Data Design & Architecture

1. Explain Kafka topics, partitions, consumer groups, offsets, and how these concepts impact throughput and scaling.
2. Kafka design question: how would you design topic partitioning for clickstream ingestion at 500k events/sec? Explain producers, consumers, and partition key trade-offs.
3. Compare Kinesis, Kafka (MSK), and SQS — when would you pick each for ingestion?
4. Change Data Capture (CDC): explain Debezium and how you would use CDC to populate a data warehouse like Snowflake or BigQuery. Discuss schema evolution and ordering guarantees.
5. Describe exactly-once, at-least-once, and at-most-once delivery semantics and give example architectures that achieve or approximate each.
6. Stream processing frameworks: compare Kafka Streams, Flink, and Spark Structured Streaming for stateful aggregations and low-latency processing.

Ankita Gulati Shubh Goyal

7. Design challenge: build a fault-tolerant, consumer-backpressure-safe pipeline from Kafka → stream processor → datastore, and explain how you'd handle reprocessing after failures.
8. Monitoring: what metrics do you track for Kafka clusters, stream processors, and consumers? How would you alert on consumer lag?
9. Security: explain how to secure Kafka (TLS, ACLs, encryption-at-rest) and considerations for multi-tenant clusters.
10. Implement a consumer that reads from Kafka, deduplicates events by event_id within a 24-hour window, and writes to a sink with idempotent writes.
11. Design an end-to-end clickstream ingestion and analytics pipeline that supports both near-real-time dashboards (≤ 30s latency) and batch analytics (daily aggregates). Provide architecture diagram, component choices, fault-tolerance, and monitoring.
12. Design a CDC system to stream changes from MySQL/Postgres into Snowflake (or BigQuery). Explain ordering, transactional guarantees, schema changes, and how you'd ensure idempotent writes.

Ankita Gulati Shubh Goyal

Thank You

Best of luck with your upcoming interviews – you've got this!

Ankita Gulati

Shubh Goyal