Data Engineer Interview Handbook: Questions & Best Answers

Table of Contents

AWS Data Engineering

Q: Explain AWS Glue architecture and how you'd use it for ETL.

A: AWS Glue is a managed ETL service offering integrated data cataloging, job orchestration, and serverless Spark processing. I use Glue to crawl, catalog, and transform data from S3, using DynamicFrames for schema flexibility. Glue jobs are triggered on schedules or events, with workflows managed via triggers or Step Functions. For example, I used Glue to process raw ingested data, automate schema detection, and load clean data to S3, leveraging partitioning and job bookmarks for incremental loads.

Q: How would you design a data lake on S3?

A: I follow Medallion Architecture with layered S3 buckets: bronze (raw), silver (cleaned), and gold (business-ready) data. Data is stored in columnar formats like Parquet, partitioned by date or business dimension for efficient access. Glue Data Catalog is used for schema management, and access is governed via IAM and Lake Formation. Lifecycle policies optimize cost by archiving old data to Glacier.

Q: Difference between EMR and Glue – when to use which?

A: Use Glue for serverless ETL, schema discovery, and small to medium Spark jobs with minimal cluster management. Choose EMR for large-scale distributed big data, advanced custom Spark or Hadoop processing, and scenarios requiring more configuration (core count, libraries, persistent clusters).

Q: Explain Lambda best practices for data processing.

A: Keep functions stateless and idempotent, use environment variables for configuration, limit payload sizes, utilize retries and dead-letter queues for error handling, and monitor with CloudWatch. Lambda is ideal for lightweight event-driven processing (S3 ingestion, notification handling).

PySpark

Q: Difference between DataFrame and RDD.

A: RDDs are the low-level API for distributed unstructured data with manual optimization, while DataFrames provide schema, optimized execution via Catalyst, and are easier for SQL-style analytics. For most data engineering, DataFrames are preferred for performance and productivity.

Q: How do you handle data skew in Spark jobs?

A: I identify skewed keys with sample aggregations, use salting to break large partitions, implement broadcast joins for small lookups, and repartition data or increase shuffle partitions as needed. Proper skew handling reduces job failures and improves runtime.

Q: What is the difference between transformations and actions in Spark?

A: Transformations (map, filter, join) are lazy, building a DAG of computation; actions (collect, count, write) trigger the computation and return results or write data.

Airflow

Q: What is a DAG and how do you define one?

A: A DAG (Directed Acyclic Graph) represents a pipeline of tasks and dependencies in Airflow. I define it in Python, setting default arguments, a schedule, and using Operators, TaskFlow API, or Sensors for each task, chaining them for dependencies.

Q: How do you handle task failures and retries in Airflow?

A: I set retries and retry_delay per task, configure email or Slack alerts, and use SLA for late tasks. Failed tasks trigger on_failure_callback for custom error handling. I also use Airflow's UI to monitor and rerun tasks as needed.

Q: What is TaskFlow API and its benefits?

A: TaskFlow API allows task definition as Python functions with type hints, improving code modularity, testability, and XCom handling. It replaces XCom with return values and provides a more Pythonic, maintainable DAG structure.

Medallion Architecture

Q: Explain Medallion Architecture and its benefits.

A: Medallion Architecture structures data in progressive layers—bronze (raw), silver (cleaned/transformed), and gold (curated for analytics). This layered approach improves data quality, auditability, modularity, and supports incremental data enrichment. I used this in my Azure Databricks project to ensure high-quality, business-ready outputs.

Q: How do you implement data quality checks in pipelines?

A: I add validation steps at the silver layer (null checks, type validation, deduplication) and assertion logic in PySpark workflows or with validation libraries like Great Expectations. Failures are logged and quarantined for analysis.

Project-Based

Q: Walk me through your Medallion Architecture implementation in Azure.

A: In my Azure E-commerce pipeline, I ingested raw sales and product data to a bronze Delta Lake, cleaned and validated it in silver using PySpark and Databricks workflows, and aggregated business metrics in gold. Parameterized Azure Data Factory pipelines orchestrated ingestion, with schema evolution and CDC handled via PySpark logic.

Q: How did you reduce regulatory submission errors by 65%?

A: I architected a Python-based validation tool using Pandas and NumPy, embedded into the batch pipeline, checking for mandatory fields, data types, and regex-based pattern matching for ratings. Errors were flagged before regulatory reporting, and I documented recurrent issues as Jira tasks for engineering fixes, leading to a 65% reduction in submission errors.

Scenario and System Design

Q: How would you migrate a legacy Informatica pipeline to AWS Glue?

A: I'd start by analyzing source and target mappings, translating transformations to Glue ETL scripts using Python or Scala, and implementing job bookmarks for incremental loads. Workflow orchestration shifts to Step Functions or Glue Workflows, and I'd ensure thorough testing and validation throughout.

Q: How would you handle a data quality issue in production?

A: Isolate the bad batch, alert stakeholders, perform root cause analysis, patch the data with backup or manual fixes, implement validation rules to prevent recurrence, and schedule an incident review with lessons learned.

SQL & Python

Q: How do you write a window function to assign a row number to each record partitioned by customer and ordered by date?

A:

```
SELECT *, ROW_NUMBER() OVER (PARTITION BY customer_id ORDER BY date_column) as rn FROM sa
```

Q: How do you read and deduplicate a CSV in Pandas?

A:

```
import pandas as pd
df = pd.read_csv('sales.csv')
df = df.drop_duplicates(['customer_id', 'order_id'])
```

Behavioral (STAR format)

Q: Tell me about a time you improved data quality standards.

A: In my regulatory reporting project, we faced frequent data submission failures due to format mismatches. I analyzed error logs, designed a validation tool in Python to check for nulls, types, and standard patterns, and integrated it pre-submission. Result: Error rates dropped by 65% and team productivity improved.

Q: Tell me about a time you delivered under tight deadlines.

A: During the Azure E-commerce pipeline project, a go-live date was moved up by two weeks. I prioritized the MVP features, coordinated daily syncs with the team, addressed blockers quickly, and worked collaboratively. We met the new deadline, and the system was stable at launch.