Databricks & Big Data Interview Q&A (2025 Edition)

Q1. What do you understand by Change Data Capture (CDC)?

Answer:
Change Data Capture (CDC) is a method to capture only the data changes (INSERT, UPDATE, DELETE) instead of reloading entire datasets.
- Ensures incremental data loading into warehouses/lakes.
- Improves performance and reduces compute/storage costs.

Use Cases:
- Real-time ETL pipelines
- Database replication to a warehouse
- Streaming ingestion with Debezium, Kafka, Azure Data Factory

Q2. Can you explain more about Databricks and its architecture?

Answer:
Databricks is a unified analytics platform built on top of Apache Spark that supports big data, ML, and AI workloads.

Architecture Components:
- Driver Node: Manages SparkContext, job scheduling.
- Worker Nodes (Executors): Perform distributed computation.
- Cluster Manager: Manages cluster lifecycle (YARN, Kubernetes, or Databricks native).
- Notebooks/UI: For interactive development, orchestration, visualization.

Use Cases: Big data ETL, ML workflows, batch/stream pipelines.

Q3. Why shouldn't we use inferred schema for large datasets?

Answer:
- Performance Overhead: Spark scans large data to guess column types.
- Inaccuracy: Sampling may assign wrong types.
- Scalability Issue: Slows initialization on big files.

Best Practice: Always define explicit schema for large/critical datasets.

Q4. What is a prolonged secrets scope in Databricks?

Answer:
A secrets scope stores credentials (keys/tokens/passwords) securely for reuse.

Types:

- Databricks-backed scopes → Managed by Databricks.
- Azure Key Vault-backed scopes → External secure integration.

Use Cases:
- Mount ADLS securely
- API/DB connections
- Secure reuse across jobs without exposing credentials

Q5. What are the parameters used while mounting storage in Databricks?

Answer:
Example:

```
dbutils.fs.mount(
  source = "wasbs://<container>@<storage>.blob.core.windows.net/",
  mount_point = "/mnt/<mount-name>",
  extra_configs = {"fs.azure.account.key.<storage>.blob.core.windows.net": "<access-key>"}
)
```

Parameters:
- source: Storage container URL
- mount_point: Path in DBFS
- extra_configs: Auth configs (Key, SAS, OAuth)

Best Practice: Store keys in secret scopes, not plain text.

Q6. Do stored procedures exist in Databricks?

Answer:
No traditional RDBMS-style stored procedures. Instead:
- Use notebooks or Delta Live Tables for procedural logic.
- Use UDFs or SQL widgets for parameterization.
- Use PySpark/Scala code for business logic.

Q7. What are different types of window functions?

Answer:
- ROW_NUMBER() – Sequential numbering.
- RANK() / DENSE_RANK() – Ranking with/without gaps.
- LAG() / LEAD() – Previous/next row values.
- NTILE(n) – Distributes rows into n buckets.
- SUM() OVER(PARTITION BY …) – Cumulative aggregations.

Use Cases: Top-N, sessionization, cumulative KPIs, time-based analysis.

Q8. What are different types of JOINs in Databricks?

Answer:
- INNER JOIN – Matching rows.

- LEFT JOIN – All from left + matches.
- RIGHT JOIN – All from right + matches.
- FULL OUTER JOIN – All rows from both.
- LEFT ANTI JOIN – Rows in left not in right.
- LEFT SEMI JOIN – Rows in left that exist in right.

Optimization: Use broadcast joins for small dimension tables.

Q9. What are UDFs in PySpark? How do they impact performance?

Answer:
A UDF (User Defined Function) lets you write custom transformations in Python/Scala.

Example:

```
from pyspark.sql.functions import udf

@udf
def upper_case(s: str) -> str:
   return s.upper()
```

Performance Considerations:
- Slower than native Spark functions.
- Break Catalyst optimizer.
- Serialization overhead (Python ↔ JVM).

Best Practice: Use built-in Spark SQL functions first.

Q10. Explain Databricks Delta Live Tables (DLT). When to use batch vs streaming?

Answer:
DLT = Managed ETL framework for reliable, declarative pipelines.

Benefits:
- Automated monitoring & data quality checks
- Schema enforcement & change propagation
- Tight integration with Delta Lake

Modes:
- Batch: For hourly/daily jobs, low-latency not required.
- Streaming: For real-time ingestion (Kafka, Event Hubs).