**11. Security & Governance**

**Scenario:**  
Your data pipeline ingests **customer PII (names, emails, phone numbers)** into Delta/S3/BigQuery. You must secure the data, comply with GDPR/CCPA, and ensure governance (who can access what, and how it’s audited).

**Baseline Approach (Expected in Interviews)**

1. **Access Control**

* Principle of least privilege: IAM roles, ACLs, Unity Catalog permissions.
* Separate roles for engineers (write) vs analysts (read-only).

1. **Encryption**

* At rest: S3 SSE, ADLS/GCS CMEK.
* In transit: TLS/SSL for Spark shuffle + JDBC/ODBC connections.

1. **PII Masking / Tokenization**

* Mask or hash sensitive columns in non-prod.
* Use pseudonymization (e.g., sha2(email)) or tokenization service.

1. **Audit & Lineage**

* Enable audit logs (cloud provider + Databricks Unity Catalog).
* Track lineage: column-level data flows (e.g., OpenLineage, Unity Catalog).

**Code snippets**

**Masking PII in Spark:**

from pyspark.sql.functions import sha2, col

df\_masked = df.withColumn("email\_hash", sha2(col("email"), 256)).drop("email")

**Column-level masking in Databricks SQL (Unity Catalog):**

CREATE FUNCTION mask\_email(email STRING) RETURNS STRING

RETURN CONCAT('\*\*\*', SUBSTRING(email, POSITION('@' IN email), LENGTH(email)));

GRANT SELECT ON TABLE customers TO analyst

MASKING POLICY mask\_email ON COLUMN email;

**Row-level security:**

CREATE ROW ACCESS POLICY region\_filter

AS (region STRING) RETURNS BOOLEAN ->

region = current\_user\_region();

ALTER TABLE sales APPLY ROW ACCESS POLICY region\_filter ON (region);

**Advanced Considerations**

* **Data retention policies:** auto-expire/delete data after X days for compliance.
* **Column-level lineage:** track transformations on PII fields (Unity Catalog, dbt, OpenLineage).
* **Differential privacy:** add noise for aggregate queries (rare but advanced).
* **Tokenization vs Encryption:**
  + Tokenization = reversible via token vault.
  + Encryption = requires key management (KMS).

**Follow-up Q&A**

**Q1. How do you ensure compliance with GDPR “right to be forgotten”?**  
👉 Maintain a delete pipeline: identify records by customer\_id → hard delete or anonymize in Bronze/Silver/Gold → propagate deletes downstream (Delta MERGE or DELETE with CDF).

**Q2. How do you secure data in non-prod environments?**  
👉 Mask/tokenize PII before copying to dev/test. Use synthetic datasets for QA.

**Q3. What’s the difference between row-level and column-level security?**  
👉 Row-level = filter rows by user (e.g., analyst only sees region=“US”).  
Column-level = hide/mask sensitive columns (analyst can’t see email).

**Q4. How do you implement data lineage?**  
👉 Tools: Unity Catalog (Databricks), OpenLineage, dbt docs. Capture transformations and dependencies automatically.

**Q5. How to enforce security in streaming pipelines?**  
👉 Apply masking during ingestion, enforce ACLs at checkpoint/output tables, encrypt Kafka topics.

**Cheat Sheet (Compressed Memory)**

* **Access:** IAM, Unity Catalog, least privilege.
* **Encryption:** SSE at rest, TLS in transit.
* **PII handling:** hash/mask/tokenize; redact in logs.
* **Row vs column security:** row = filter, column = mask.
* **Compliance:** right-to-be-forgotten → delete/anonymize with CDC.
* **Audit:** log access + lineage.
* **Non-prod:** synthetic or masked data only.