**Understanding Data Provenance & Lineage**

**What is Data Provenance?**

**Definition:**Data provenance refers to the history, origin, and transformation of data over time. It answers:

* Where did this data come from?
* Who created or modified it?
* What operations were performed on it?
* When and how was it changed?

Think of it as a “data passport” that records every checkpoint in the lifecycle of a dataset.

**Key Aspects of Provenance:**

1. **Source Information – Original creator/system of the data.**
2. **Ownership & Custodianship – Who is responsible for it at each stage.**
3. **Processing History – What ETL/ML/analytical steps were applied.**
4. **Metadata – Contextual information (timestamps, version, tools used).**
5. **Auditability – Ability to trace back data origin for compliance and trust.**

**Example:  
In a banking application, a "Customer Balance" column may have provenance metadata showing:**

* **Extracted from transactions DB on 2025-09-24 at 10:00 AM**
* **Cleaned (nulls replaced with 0)**
* **Aggregated via Spark job balance\_agg\_v2**
* **Loaded into reporting warehouse**

**What is Data Lineage?**

**Definition:  
Data lineage focuses on how data flows through systems: from source → transformations → destination. It shows relationships between datasets and the path data travels.**

**Think of it as a map of data pipelines.**

**Key Aspects of Lineage:**

1. **End-to-End Flow – Movement from raw sources to final reports/ML models.**
2. **Transformation Mapping – Joins, filters, aggregations, calculations.**
3. **Impact Analysis – If source changes, which reports/models will break?**
4. **Regulatory Transparency – Proves how results (like credit score, loan decision) were computed.**

**Example:  
An ML model predicting insurance risk may use lineage to trace:**

* **Data from Policy DB → cleaned via Python script → enriched with demographics API → fed into feature engineering pipeline → used in XGBoost model for predictions.**

**Provenance vs Lineage**

| **Feature** | **Data Provenance** | **Data Lineage** |
| --- | --- | --- |
| **Focus** | **History & origin of data** | **Movement & transformations of data** |
| **Granularity** | **More detailed (who, when, how)** | **More structural (how data flows)** |
| **Use Case** | **Trust, reproducibility, audit** | **Debugging, impact analysis, governance** |
| **Analogy** | **Passport (tracks identity + history)** | **Map (tracks journey + routes)** |

**Why Important?**

1. **Data Quality – Ensures trust in insights.**
2. **Compliance – GDPR, HIPAA, RBI require traceability.**
3. **Reproducibility – Critical in ML experiments.**
4. **Impact Analysis – Quick troubleshooting if pipelines break.**
5. **Security & Governance – Who accessed or modified sensitive data.**

**Real-World Examples**

* **Healthcare: Provenance ensures patient diagnosis data is traceable to labs & doctors.**
* **Finance: Lineage maps how raw transactions become regulatory reports.**
* **AI/ML: Provenance guarantees models are explainable (responsible AI).**

**In short:**

* **Data Provenance = *History + Identity of data***
* **Data Lineage = *Flow + Path of data***

**They complement each other: provenance gives detail, lineage gives structure.**

**Data Provenance**

* The **origin/history** of the data.
* Answers: *Where did this data come from? How was it collected?*
* Example (Banking): A customer’s loan record comes from the *Loan Management System*, extracted on *2025-09-23*.

**Data Lineage**

* The **journey of the data** across systems and transformations.
* Answers: *How did this data move and transform across pipelines?*
* Example (Banking): Customer transaction data → cleaned (remove duplicates) → aggregated (monthly total) → stored in Data Warehouse → used in Credit Risk ML model.

**Why It Matters in Banking?**

* **Auditability** → Regulators (RBI, SEC) demand full traceability of financial data.
* **Trust** → Business decisions rely on knowing data wasn’t corrupted/altered.
* **Debugging** → If fraud detection model gives wrong results, lineage helps track where data went wrong.

**Tracking Metadata & Data Flow for Auditability**

Metadata you might track in banking pipelines:

* **Source** → Core Banking System, ATM Switch, Mobile App Logs
* **Timestamp** → When was data extracted/loaded
* **Transformation** → Aggregation, normalization, filtering
* **Owner** → Who handled the data (analyst/system)
* **Version** → Schema or pipeline version

**Design and implement an ETL (Extract–Transform–Load) pipeline for an e-commerce system.**

The pipeline should:

1. **Extract**
   * Fetch product and order data from a given **REST API** (JSON format).
   * Support an additional **EDI file input** (simulating supplier/warehouse data).
2. **Transform**
   * Parse JSON and EDI formats into a common internal structure.
   * Handle missing/invalid fields gracefully.
   * Enrich data (e.g., calculate order total, normalize product categories).
3. **Load**
   * Store the cleaned and transformed data into a structured format (CSV or relational database like PostgreSQL/MySQL).
4. **Error Handling & Reliability**
   * Implement **retry logic** when API requests fail.
   * Add **logging** for each ETL stage (success, warnings, and errors).
   * Ensure idempotency (running the pipeline twice should not create duplicates).
5. **Deliverables**
   * Source code (Python/Java/Scala or your preferred language).
   * Configurable pipeline (API endpoint, file path, database connection via config file).
   * A sample run with logs and final structured output.