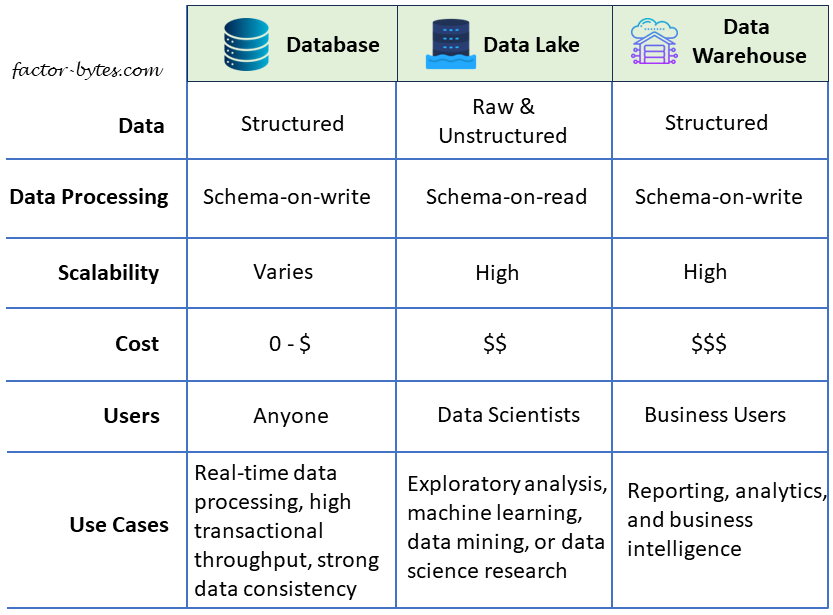
# **🌊 Day 15 : Data Lake & Delta Lake in Action 📅 Date:** 27/06/2025

****

## **🔹 1. What is a Data Lake?**

A **Data Lake** is a centralized repository that lets you store **structured, semi-structured, and unstructured data** at any scale. Unlike traditional databases, it supports **schema-on-read**, which means the schema is applied only during query time, making it very flexible and scalable.

### 

### **Features:**

* **Stores all data types**: CSV, JSON, images, logs, Parquet, videos, etc.
* **Scalable**: From GBs to petabytes.
* **Cost-efficient**: Stores raw data in cheap cloud storage like Azure Data Lake Storage Gen2.
* **Decoupled compute & storage**: Databricks clusters can read without moving data.

**Medallion Architecture in Data Lakes**

The **Medallion Architecture** organizes data into layers for better governance and performance:

| **Layer** | **Description** |
| --- | --- |
| 🥉 Bronze | Raw ingested data (CSV, JSON, etc.) |
| 🥈 Silver | Cleaned, type-casted, conformed data |
| 🥇 Gold | Aggregated and business-ready insights |

This approach is **modular, scalable, and reliable** for building production-grade data pipelines.

## **What is Delta Lake?**

**Delta Lake** is a storage layer built on top of Apache Spark and Parquet that brings **ACID transactions, time travel, schema evolution**, and **streaming support** to your data lake.

**Why Delta Lake?**

* Prevents **data corruption** during parallel writes
* Allows **updates, deletes, and merges** like SQL
* Supports **versioning** (time travel)
* Enables **real-time + batch processing** on the same table

### **Key Features:**

| **Feature** | **Description** |
| --- | --- |
| ACID Transactions | Ensures consistent reads/writes |
| Schema Enforcement | Validates schema at write time |
| Schema Evolution | Add new columns without rewrite |
| Time Travel | Query older versions of the table |
| Upserts (MERGE INTO) | Insert/update based on conditions |
| Unified Batch/Stream | Supports both types of workloads |
| Optimize & Z-Order | Boosts performance by file compaction & column index |

## **Hands-on Project: Lending Club Loan EDA with Medallion Architecture**

### **📂 Dataset Used:**

* **File:** accepted\_2007\_to\_2018Q4.csv (~1.7 GB)
* **Source:** [Lending Club Loan Dataset](https://www.lendingclub.com/info/download-data.action)

### **Bronze Layer – Raw Ingestion**

Notebook: BronzeLayer

file\_path = "/FileStore/shared\_uploads/azuser3611\_mml.local@techademy.com/accepted\_2007\_to\_2018Q4.csv"

df\_bronze = spark.read.option("header", True).option("inferSchema", True).csv(file\_path)

df\_bronze.write.format("delta").mode("overwrite").save("/FileStore/lending/bronze/delta")

### **Silver Layer – Data Cleaning & Transformation**

Notebook: SilverLayer

from pyspark.sql.functions import col, regexp\_replace

df\_bronze = spark.read.format("delta").load("/FileStore/lending/bronze/delta")

df\_silver = df\_bronze.dropna(subset=["loan\_amnt", "term", "int\_rate"])\

.withColumn("int\_rate", regexp\_replace("int\_rate", "%", "").cast("double"))\

.withColumn("loan\_amnt", col("loan\_amnt").cast("double"))\

.withColumn("annual\_inc", col("annual\_inc").cast("double"))\

.withColumn("emp\_length", regexp\_replace("emp\_length", "\\+ years| years|< 1 year", ""))\

.withColumn("emp\_length", regexp\_replace("emp\_length", "n/a", "0").cast("int"))

df\_silver.write.format("delta").mode("overwrite").save("/FileStore/lending/silver/delta")

### **Gold Layer – Aggregated Business Insights**

Notebook: GoldLayer

from pyspark.sql.functions import avg

df\_silver = spark.read.format("delta").load("/FileStore/lending/silver/delta")

loan\_summary = df\_silver.groupBy("grade").agg(

avg("loan\_amnt").alias("avg\_loan"),

avg("int\_rate").alias("avg\_interest")

)

loan\_summary.write.format("delta").mode("overwrite").save("/FileStore/lending/gold/loan\_summary\_by\_grade")

**Delta Lake SQL in Action**

| **Action** | **SQL Example** |
| --- | --- |
| Create Table | CREATE TABLE sales\_delta USING DELTA AS SELECT \* FROM raw\_sales\_data |
| Insert | INSERT INTO sales\_delta VALUES (...) |
| Update | UPDATE sales\_delta SET amount = 550 WHERE id = 1001 |
| Delete | DELETE FROM sales\_delta WHERE id = 1001 |
| Merge (Upsert) | MERGE INTO ... WHEN MATCHED THEN UPDATE ... |
| Time Travel | SELECT \* FROM sales\_delta VERSION AS OF 3 |
| Optimize | OPTIMIZE sales\_delta ZORDER BY (date) |