**Case Study: Building a customer 360 Analytics Platform with Delta Lake**

**Business Scenario**

A retail company, **RetailMart**, operates both online and offline stores across multiple regions.  
They want to build a **customer 360 Analytics Platform** to answer critical business questions:

* Who are our top customers by revenue?
* How do customer preferences change over time?
* Can we track address or profile changes while preserving historical data?
* How do we support both **real-time dashboards** and **machine learning models** (like churn prediction)?

Traditionally, RetailMart used a **data lake (Parquet/CSV on Azure Data Lake Storage)**. However, they faced issues:

* Difficulty managing updates/deletes (slowly changing dimensions).
* Inconsistent reads when multiple jobs wrote data simultaneously.
* Hard to enforce schemas (bad/malformed data entered the lake).
* Complex ETL pipelines to handle both batch and streaming.

To solve this, RetailMart adopted **Delta Lake** on **Azure Databricks** as the foundation of their **Lakehouse** architecture.

**Architecture Overview**

1. **Data Sources**
   * Online orders (JSON stream from Kafka/Event Hub).
   * Store transactions (CSV dumps from POS systems).
   * Customer master data (Parquet/CSV from CRM).
2. **Ingestion Layer**
   * Batch loads from CRM and POS → Raw Zone (ADLS Gen2).
   * Streaming ingestion from Kafka → Bronze Delta tables.
3. **Processing Layer (Delta Lake)**
   * **Bronze:** Raw ingested data.
   * **Silver:** Cleaned, merged, schema-enforced Delta tables.
   * **Gold:** Aggregated tables (customer 360, revenue reports).
4. **Consumption Layer**
   * Power BI for dashboards.
   * ML models (customer churn, recommendations) using Spark ML/MLflow.

**Datasets**

**1. Customers (CRM)**

| **id** | **name** | **email** | **address** | **join\_date** |
| --- | --- | --- | --- | --- |
| 1 | Alice | alice@xyz.com | NY | 2020-01-01 |
| 2 | Bob | bob@xyz.com | LA | 2021-03-15 |
| 3 | Charlie | charlie@xyz.com | Chicago | 2022-05-20 |

**2. Orders (POS & Online)**

| **order\_id** | **customer\_id** | **amount** | **order\_date** | **channel** |
| --- | --- | --- | --- | --- |
| 101 | 1 | 250 | 2025-09-01 | Online |
| 102 | 2 | 180 | 2025-09-01 | Store |
| 103 | 1 | 300 | 2025-09-02 | Online |

**3. Streaming Orders (JSON via Kafka/Event Hub)**

{"order\_id":201,"customer\_id":3,"amount":99,"order\_date":"2025-09-05","channel":"Store"}

{"order\_id":202,"customer\_id":2,"amount":75,"order\_date":"2025-09-05","channel":"Online"}

**Hands-On Labs in Case Study**

**Lab 1: Data Ingestion**

* Ingest **customers.csv** into Bronze Delta.

df = spark.read.option("header","true").csv("/mnt/raw/customers.csv")

df.write.format("delta").save("/mnt/bronze/customers")

**Lab 2: Schema Enforcement**

* Try writing a mismatched schema (extra column).

bad\_df = spark.createDataFrame([(1,"Alice","NY",30)], ["id","name","address","age"])

bad\_df.write.format("delta").mode("append").save("/mnt/bronze/customers")

# → Fails due to schema enforcement

**Lab 3: SCD Type 2 with MERGE**

* Update customer address but preserve history.

MERGE INTO customers\_silver t

USING updates s

ON t.id = s.id AND t.current = true

WHEN MATCHED AND t.address <> s.address

THEN UPDATE SET t.current = false, t.end\_date = current\_date()

WHEN NOT MATCHED

THEN INSERT (id,name,address,start\_date,end\_date,current)

VALUES (s.id,s.name,s.address,current\_date(),NULL,true);

**Lab 4: Time Travel**

* Query customer data as of last week.

spark.read.format("delta").option("versionAsOf",1).load("/mnt/silver/customers").show()

**Lab 5: Performance Optimization**

* Optimize Orders Delta table with Z-Ordering.

OPTIMIZE orders\_silver ZORDER BY (customer\_id);

**Lab 6: Streaming Orders**

* Write Kafka JSON stream to Delta.

stream\_df = spark.readStream.format("json").schema(schema).load("/mnt/stream/orders")

stream\_df.writeStream.format("delta").option("checkpointLocation","/mnt/checkpoints/orders") \

.outputMode("append").start("/mnt/bronze/orders\_stream")

**Lab 7: BI Consumption**

* Create Gold Layer for reporting.

CREATE TABLE customer\_360 AS

SELECT c.id, c.name, c.email, SUM(o.amount) as total\_spent

FROM customers\_silver c

JOIN orders\_silver o ON c.id = o.customer\_id

GROUP BY c.id, c.name, c.email;

* Connect Power BI / Synapse Serverless SQL to customer\_360.

**Expected Outcomes**

* Reliable **Customer 360 dataset** with history tracking.
* Unified **batch + streaming ingestion** in one platform.
* Ability to perform **time travel queries** for audits.
* Faster BI dashboards due to Delta optimizations.
* A scalable **Lakehouse architecture** built on Delta Lake.