Case Study- HDFC

Case Study 1: Retail Loan Disbursement & Customer 360 Lakehouse Pipeline

Business Context

HDFC Bank operates separate systems for customer onboarding, loan origination, disbursement processing, and branch operations. Each system produces daily batch extracts in different formats and structures.
Due to **schema mismatches, duplicate customers, partial records, and delayed disbursement updates**, business teams lack a single, trusted view of customer loan exposure and branch performance.

Problem Statement

Design and implement a **production-grade batch data pipeline** that consolidates customer, loan, and disbursement data into a governed Lakehouse architecture, ensuring data quality, auditability, and analytics readiness.

Objectives

- Standardize heterogeneous banking datasets into a unified data model
- Eliminate duplicates and inconsistencies using deterministic logic
- Apply business rules for loan lifecycle and customer risk profiling
- Build Bronze, Silver, and Gold layers aligned with enterprise standards
- Ensure data governance, validation, and traceability

Detailed Tasks

Data Ingestion & Storage (Azure)

1. Design ADLS Gen2 folder structure (landing / bronze / silver / gold)
2. Configure ADF linked services for:
    - Source systems
    - ADLS Gen2
    - Azure Databricks
3. Create parameterized ADF datasets for multi-format ingestion
4. /)

Bronze Layer Processing (PySpark)

5. Read CSV, JSON, and Parquet files into DataFrames
6. Apply explicit schema definitions to avoid inference issues
7. Persist raw ingested data as **Bronze Delta tables**
8. Capture ingestion timestamps and source identifiers

Silver Layer Transformation (PySpark + SQL)

9. Perform null handling and mandatory field validation
10. Standardize data types (dates, amounts, identifiers)
11. Remove duplicate customers using:
    - Window functions (`row_number` over customer_id, update_ts)
12. Resolve multiple loan records per customer
13. Apply joins across customer, loan, disbursement, and branch tables
14. Create derived columns:
    - Loan age
    - Disbursement delay
    - Customer risk band (UDF)
15. Optimize transformations using partitioning and caching

Gold Layer & Governance

16. Design analytics-friendly Gold schemas
17. Create managed Delta tables using **Unity Catalog**
18. Create external tables for BI tool access
19. Build SQL views for reporting use cases
20. Apply validation checks (counts, totals, reconciliation)

Outcomes

- A **Customer 360 Gold dataset** aligned to banking analytics needs
- Branch-level loan disbursement KPIs
- Governed, versioned Delta tables with auditability
- Practical understanding of **enterprise batch pipelines in BFSI**

Case Study 2: Credit Card Transaction Processing & Incremental Analytics

Business Context

HDFC Bank processes millions of credit card transactions daily from POS, online, and international channels. Data arrives in **incremental batches** with frequent duplicates, late-arriving records, and changing schemas.

Problem Statement

Build an **incremental transaction processing pipeline** that guarantees correctness, supports historical updates, and prepares datasets for fraud and risk analytics.

Objectives

- Implement reliable incremental ingestion using Azure Data Factory
- Handle late-arriving and duplicate transaction data
- Maintain transaction history using Delta Lake capabilities
- Produce daily and monthly transaction aggregates

Detailed Tasks

Incremental Ingestion (ADF)

1. Design watermark-based ingestion strategy
2. Parameterize ADF pipelines for date-based processing
3. Archive processed files to ensure idempotency

Transaction Processing (PySpark)

4. Read incremental files into DataFrames
5. Standardize timestamps, currency, and transaction amounts
6. Identify duplicates using:
    - Card number
    - Merchant ID
    - Transaction timestamp

7. Deduplicate using window functions
8. Join with card master and merchant reference datasets
9. Implement **MERGE INTO Delta** for upserts
10. Handle schema evolution safely

Analytics & Validation

11. Create SQL views for:
    - Daily transaction volume
    - Spend by merchant category
12. Perform anomaly detection using subqueries
13. Generate summaries using Pandas and NumPy
14. Validate totals against source system extracts

Outcomes

- Incrementally updated, fraud-ready transaction datasets
- Accurate daily aggregates with audit history
- Hands-on experience with **Delta Lake MERGE patterns**
- Understanding of banking transaction data challenges

Case Study 3: Branch Performance & Operations Analytics Platform

Business Context

Senior leadership at HDFC Bank requires a **single source of truth** for branch performance across products, balances, and customer activity. Existing reports are slow and manually reconciled.

Problem Statement

Design an analytics platform that consolidates operational data into **BI-optimized Gold tables** with strong governance and validation.

Objectives

- Centralize operational datasets into a Lakehouse
- Create analytics-optimized Gold datasets
- Enable fast, reliable BI reporting
- Implement strong validation and reconciliation logic

Detailed Tasks

Data Orchestration

1. Orchestrate ingestion dependencies using ADF
2. Ensure consistent cut-off times across datasets

Transformation & Aggregation (PySpark)

3. Join customer, account, product, and branch datasets
4. Use window functions to derive latest balances
5. Aggregate metrics by:
    - Branch
    - Product
    - Date
6. Persist Gold Delta tables with optimization
7. Register datasets in Unity Catalog

Reporting Enablement

8. Create SQL views for BI tools
9. Implement date-based trend analysis
10. Generate executive summaries using Python
11. Validate KPIs against finance extracts

Outcomes

- Enterprise-grade branch performance datasets
- BI-ready views for dashboards
- Confidence in leadership reporting accuracy
- Real-world exposure to analytics platform design
