BIG DATA IN FINANCE AND BANKING:

CAT 1: DSA 8504

11/02/2025

Lecturer: Dr. John Olukuru | Assistant Instructor: Joan Ngugi

This Takeaway CAT is divided into two sections:

1. **Section 1**: Read the case study carefully and answer the questions, making specific references to the details provided in the case(30 Marks)
2. **Section 2**: Complete the practical implementation task as instructed. Please submit this section in **notebook format** (70 Marks)

Submission Deadline Date : 23/02/2025 – 9pm

Section 1: Case Study(30 Marks)

HakiLend’s Big Data Modernization Project

Context & Unique Constraints

HakiLend is a microfinance institution focused on providing consumer loans across **20 countries** in East and West Africa. Recently, the bank’s CTO approved a **modernization project** to build a new Big Data platform. This platform aims to integrate **historical batch data**, **real-time transaction streams**, and **third-party credit risk data**.

1. **Current Technology & Pain Points**
   - **Legacy Systems**: Transactional data (20 years’ worth) is stored on a mainframe using COBOL-based applications.
   - **Volume Explosion**: HakiLend processes **20 million transactions daily**, largely due to the rise of mobile banking.
   - **Emerging Markets**: Expansion into two new countries with **strict data sovereignty laws** requires in-country data storage and specific encryption standards.
   - **Limited Development Team**: HakiLend’s in-house Big Data team is small, seeking **straightforward, cost-effective** solutions that can scale as hiring increases.
2. **Business & Compliance Drivers**
   - **Immediate Fraud Detection**: Near real-time alerts for suspicious cross-border credit card transactions.
   - **Regulatory Audits**: Multiple regulators (including **GDPR** and local authorities) require data retention, audit trails, and granular access control.
   - **Credit Risk Analytics**: HakiLend receives **semi-structured** credit rating data from two external agencies, which must be merged with internal historical data for advanced modeling.
   - **Cost Concerns**: The CTO demands a **high-level cost estimate** and is open to **cloud services** if they reduce long-term expenses without sacrificing data control.

3. **High-Level Goals**
    - **Unified Data Environment**: Capable of storing **structured**, **semi-structured**, and **unstructured** data (or at least seamlessly integrating them).
    - **Modular Big Data Architecture**: Clear layers for ingestion, storage, batch processing, real-time streaming, and governance.
    - **Scalable Analytics**: Must support **machine learning** (e.g., credit risk scoring, customer behaviour analysis).

TASKS & MARK DISTRIBUTION

Total Marks: **30**

Instructions

1. Please answer **all** questions.
2. **Length & Format**: Each response should be concise but sufficiently detailed. You may include diagrams where necessary.

---

1. Introduction to Big Data (2 Marks)

- In your own words, **justify why HakiLend's scenario qualifies as a Big Data challenge.**Reference the **3 Vs** (Volume, Velocity, Variety) (and others if relevant) to support your argument.

2. Big Data Architecture & Components (13 Marks)

Task:

1. **Propose a high-level, end-to-end architecture** for HakiLend, including batch and streaming layers.
2. Provide a **diagram** labeling key components (e.g., Hadoop, Spark, or cloud services).
3. **Explain** how each component addresses HakiLend's needs (legacy integration, real-time fraud detection, data governance, etc.).

3. Common Big Data Challenges in Banking (3 Marks)

Task:

1. Discuss **two typical challenges** (technical, cultural, or regulatory) that banks face when implementing Big Data solutions.
2. For each challenge, provide **one feasible solution** relevant to HakiLend's **limited in-house Big Data team**.

4. HakiLend's Justification (2 Marks)

Task:

1. Summarize **why a traditional relational database** alone is **insufficient** for HakiLend's scenario.
2. Explain **why** a shift to modern Big Data solutions is **inevitable** for long-term competitiveness.

6. YARN & Resource Management (2 Marks)

Scenario: HakiLend wants multiple teams (fraud analytics, risk modeling, marketing) to run Hadoop jobs **simultaneously** on a shared cluster.

Task:

- **Explain how YARN (Yet Another Resource Negotiator)** allocates cluster resources among different applications.
- Mention **any key features** that ensure balanced resource usage

7. Hadoop Ecosystem Tools (2 Marks)

Task:

1. Identify **two Hadoop ecosystem tools** (beyond HDFS and MapReduce) that might benefit HakiLend.
2. For each tool, describe a **use case** specific to HakiLend (e.g., data ingestion, analytics, workflow orchestration).

8. RDDs, DataFrames, and Datasets (3 Marks)

Context: HakiLend's data engineering team is migrating batch jobs from MapReduce to Spark.

Task:

1. Compare **RDDs**, **DataFrames**, and **Datasets** in Spark.
2. **Recommend** which API you would use for **complex ETL on structured financial data**, justifying your choice.

9. Tool Selection & Complexity (2 Marks)

Task:

- In your own words, **explain how you would streamline tool selection** to ensure HakiLend's **small team** can manage the Big Data environment effectively.

Section 2: Practical Implementation (70 Marks)

Optimizing Collaborative Filtering with Spark ALS

Background

In class, we learned about **Collaborative Filtering** for recommendation systems using **Singular Value Decomposition (SVD)**. We applied this technique on the **transactions.csv** dataset, which contains **1 million records**.

However, **SVD was inefficient for this dataset**:

- It took **too long to run** due to large matrix computations.
- It was **not optimized for big data** since it processed everything in memory.

Task Overview

To **improve performance**, we will use **Apache Spark** to:

✓ **Load & preprocess the dataset using PySpark** (instead of Pandas).

✓ **Perform basic EDA to understand the dataset more using PySpark**

✓ **Apply ALS (Alternating Least Squares)** for collaborative filtering (instead of SVD).

- ✓ **Generate top-N product recommendations for users efficiently**.

Why ALS instead of SVD?

SVD is slow for large, sparse user-item matrices.

- It performs **matrix factorization on the entire dataset**, which becomes computationally expensive.
- It does **not scale well for large datasets**.

ALS (Alternating Least Squares) is optimized for large-scale collaborative filtering.

- It **runs efficiently on distributed Spark clusters**.
- It **converges faster than SVD**, especially for sparse recommendation matrices.

Assignment Expectations

- Complete an end-to-end PySpark ALS implementation (hint :Use Google Collab if PySpark installation is not working on local machine)
- Preprocess & split the dataset correctly
- Train & evaluate an ALS model (RMSE score must be reported)
- Generate top-3 recommendations for each user as a dataframe

Deliverables

- ✓ Python script (.py or .ipynb) implementing ALS-based recommendations.
- ✓ RMSE score on test data.
- ✓ Generated recommendations for test users saved to a csv file.