|  |
| --- |
|  |

  **Project Document**

**Project Title: Money Laundering Detection System – Banking**

**1. Introduction**

Money laundering is one of the biggest challenges in the financial sector. Criminals use illegal money and move it across accounts in small amounts to avoid detection. To prevent this, banks need automated systems that can handle large amounts of transaction data and identify suspicious activity.

This project aims to design a **Big Data ETL pipeline using Scala and Apache Spark** that ingests transaction data, processes it, stores it in a structured format, and helps detect anomalies (potential money laundering cases).

**2. Objectives**

* Ingest transaction data from multiple sources (CSV/dummy data instead of real Kafka for simplicity).
* Process and clean the data using **Scala + Spark**.
* Model the data in a **Star Schema** for easy querying.
* Identify suspicious transactions using simple anomaly detection rules.
* Store results in a data warehouse for reporting.

**3. Tools & Technologies**

* **Programming:** Scala
* **Processing:** Apache Spark (batch, not streaming for simplicity)
* **Storage:** HDFS (or local file system if HDFS unavailable)
* **Database:** PostgreSQL (for structured warehouse storage)
* **Data Modeling:** Star schema (fact + dimension tables)
* **Analytics:** Simple Spark queries (instead of a full BI dashboard)

**4. Data Description**

We will use **dummy banking transaction data**.

**Fact Table: fact\_transactions**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Type** | **Description** |
| transaction\_id | String | Unique transaction identifier |
| customer\_id | String | Customer involved |
| account\_id | String | Bank account used |
| amount | Double | Transaction amount |
| transaction\_type | String | Credit/Debit |
| timestamp | Timestamp | Date & time of transaction |

**Dimension Tables**

* **dim\_customer** (customer\_id, name, location, risk\_category)
* **dim\_account** (account\_id, customer\_id, account\_type, branch)
* **dim\_date** (date\_id, day, month, year, quarter)

**5. Project Workflow**

**Step 1: Data Ingestion**

* Load transaction data (CSV file) into Spark using Scala.
* Store raw data in HDFS (or local).

val transactions = spark.read.option("header","true")

 .csv("transactions.csv")

**Step 2: Data Cleaning & Transformation**

* Convert columns into correct datatypes.
* Create derived column: is\_suspicious.

Suspicious rules (beginner level):

* Transaction amount > 10,000
* More than 5 transactions in 10 minutes by same customer

import org.apache.spark.sql.functions.\_

val cleanedDF = transactions

 .withColumn("amount", col("amount").cast("double"))

 .withColumn("timestamp", col("timestamp").cast("timestamp"))

 .withColumn("is\_suspicious", when(col("amount") > 10000, 1).otherwise(0))

**Step 3: Data Modeling (Star Schema)**

* Separate data into **fact\_transactions** and **dimension tables**.
* Save in Parquet format for efficiency.

cleanedDF.write.mode("overwrite").parquet("fact\_transactions")

**Step 4: Data Warehousing**

* Load fact and dimension tables into PostgreSQL for querying.
* Store suspicious transactions separately in MongoDB (optional, but can be skipped for simplicity).

**Step 5: Analytics**

Sample queries:

* **Find suspicious transactions**

val suspicious = cleanedDF.filter(col("is\_suspicious") === 1)

* **Top customers with most suspicious activity**

val topCustomers = suspicious.groupBy("customer\_id")

 .count()

 .orderBy(desc("count"))

* **Monthly transaction volume**

val monthlyTxn = cleanedDF.groupBy(month(col("timestamp")).alias("month"))

 .agg(sum("amount").alias("total\_amount"))

**6. Expected Output**

* Processed fact and dimension tables stored in warehouse.
* List of suspicious transactions flagged for investigation.
* Reports like:
  + Top suspicious customers
  + High-value transactions
  + Transaction trends over time

**7. Challenges & Solutions**

* **Large data size** → Use Spark distributed processing.
* **Data inconsistency** → Clean & validate in ETL stage.
* **Storage optimization** → Use Parquet instead of raw CSV.

**8. Future Enhancements**

* Integrate **Kafka + Spark Structured Streaming** for real-time ingestion.
* Use **Spark MLlib** for advanced anomaly detection models.
* Build a **BI dashboard** (Tableau/PowerBI) for AML officers.

**9. Conclusion**

This project provides a beginner-friendly **Big Data Engineering pipeline in Scala** for detecting suspicious financial transactions. While simplified, it captures the **core steps of ingestion, processing, modeling, and analytics**, and can be extended for real-world complexity.