**Report on Health Management System**

**Introduction**

The Health Data Analysis project leverages Apache Spark for large-scale data processing and MongoDB for efficient data storage. The system generates and analyzes synthetic patient health records, extracting insights while ensuring scalability and reliability. This report provides a structured overview of the implementation, focusing on data generation, processing, and storage.

**Objectives**

* Generate a dataset of 10,000 synthetic patient health records.
* Load the dataset into Apache Spark for analysis.
* Perform basic statistical analysis on health metrics.
* Store patient feedback in MongoDB for future reference.
* Ensure seamless integration between Spark and MongoDB.

**System Architecture**

The system consists of the following components:

1. **Data Generation**: The script creates synthetic patient records with randomly assigned values for blood pressure, glucose levels, lipid levels, and hemoglobin count.
2. **Apache Spark Processing**: The generated dataset is loaded into a Spark DataFrame for structured querying and statistical computations.
3. **MongoDB Integration**: Patient feedback is stored in MongoDB for further analysis and retrieval.

**Implementation**

**Step 1: Data Generation**

The script generates 10,000 synthetic patient records with realistic values for key health metrics.

import random

import json

def create\_records(count=10000):

return [

{

"record\_id": i,

"blood\_pressure": random.randint(90, 180),

"glucose": round(random.uniform(70, 250), 1),

"lipid\_levels": random.randint(100, 300),

"hb\_count": round(random.uniform(9, 16), 1)

}

for i in range(count)

]

data = create\_records()

with open("health\_records.json", "w") as file:

json.dump(data, file)

This block of code simulates real-world medical data, ensuring diversity in patient health conditions.

**Step 2: Data Processing with Apache Spark**

Apache Spark loads and processes the generated dataset.

from pyspark.sql import SparkSession

session = SparkSession.builder.master("local[\*]").appName("HealthDataAnalysis").getOrCreate()

dataframe = session.read.option("multiline", "true").json("health\_records.json")

dataframe.createOrReplaceTempView("medical\_records")

A temporary SQL table is created to facilitate structured queries on patient data.

**Step 3: Statistical Analysis**

The script computes average values for various health indicators.

metrics = session.sql("""

SELECT

AVG(blood\_pressure) as avg\_bp,

AVG(glucose) as avg\_glucose,

AVG(lipid\_levels) as avg\_cholesterol,

AVG(hb\_count) as avg\_hb

FROM medical\_records

""")

metrics.show()

This query retrieves key insights, such as the average glucose level and hemoglobin count, helping healthcare professionals identify trends.

**Step 4: MongoDB Integration**

MongoDB stores patient feedback for future reference.

from pymongo import MongoClient

try:

db\_client = MongoClient('localhost', 27017, serverSelectionTimeoutMS=5000)

db\_client.server\_info()

health\_db = db\_client['medical\_db']

feedback\_table = health\_db['patient\_feedback']

print("Connected to database successfully.")

except Exception as err:

print("Database connection failed:", err)

**Step 5: Storing Patient Feedback**

def log\_feedback(entry\_id, remark):

if db\_client:

feedback\_table.insert\_one({"record\_id": entry\_id, "response": remark})

print(f"Feedback logged for Record {entry\_id}: {remark}")

log\_feedback(150, "Positive response")

log\_feedback(275, "Needs follow-up")

This enables tracking patient satisfaction and medical outcomes.

**Results and Observations**

* **Data Integrity**: The generated dataset provides a realistic simulation of patient records.
* **Scalability**: Spark's distributed computing capabilities ensure smooth processing of large datasets.
* **Insight Extraction**: SQL-based analysis identifies patterns in patient health metrics.
* **Data Persistence**: MongoDB efficiently stores and retrieves patient feedback.

**Conclusion**

This project demonstrates a seamless workflow for generating, processing, analyzing, and storing healthcare data. The integration of Apache Spark and MongoDB provides a scalable and efficient framework for real-time analytics. Future improvements may include real-time data streaming and machine learning-based patient risk assessment.