Sauda Haywood

DSC650

Week 11-12: Final Project

**Amazon Stock Data Pipeline Report**

**Introduction**

This report details the end-to-end data pipeline for analyzing Amazon stock data using NiFi, HDFS, Hive, and PySpark. The project involves collecting historical stock data from [Nasdaq](https://www.nasdaq.com/market-activity/stocks/amzn/historical), storing it in HDFS, processing it using Hive, and performing analysis with PySpark.

**Dataset**

* **Dataset Source:** [Nasdaq - Amazon Stock Historical Data](https://www.nasdaq.com/market-activity/stocks/amzn/historical)
* **GitHub Repository:** [Amazon Stock Data](https://github.com/saudahaywood/amazonstock/blob/main/amazonstock.csv)
* The dataset contains historical stock prices, including trade date, closing price, volume, open price, high, and low prices.

**Pipeline Overview**

The data pipeline follows these steps:

1. **NiFi** pulls the dataset from the GitHub repository.
2. **HDFS** stores the raw ingested data.
3. **Hive** creates structured tables to allow querying.
4. **PySpark** performs transformations and analysis.

**Issues Encountered and Resolutions**

**1. Issues with Data Flow Connection**

* **Problem:** Difficulty in connecting the flow and retrieving data from NiFi to HDFS and Hive.
* **Cause:** Missing Hadoop Configuration Resources.
* **Solution:** Added the correct Hadoop configuration resources in NiFi to resolve the issue.

**2. NiFi, Hadoop, and VM Crash**

* **Problem:** NiFi, Hadoop, and all components stopped working unexpectedly.
* **Cause:** The VM crashed due to excessive file ingestion from an unchecked data flow.
* **Solution:** Recreated the VM using the tutorial from Week 1 and Run Once the flow to prevent excessive data flow.

**3. Concurrent Execution Issues**

* **Problem:** Errors when opening Hadoop and NiFi simultaneously.
* **Cause:** Resource allocation issues causing conflicts between the two systems.
* **Solution:** Ensured that I was running one system at time.

**4. Errors in Table Creation and Queries**

* **Problem:** Issues with table creation and running Hive queries.
* **Cause:** Syntax errors.
* **Solution:** Edited and corrected the syntax errors.

**Code**

**1. Hive**

**Creating Database:**

CREATE DATABASE final\_project;

**Creating Tables:**

USE final\_project;

CREATE EXTERNAL TABLE amazon\_stocks (

trade\_date STRING,

close\_last STRING,

volume STRING,

open\_price STRING,

high STRING,

low STRING

)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

STORED AS TEXTFILE

LOCATION '/AmazonStocks'

TBLPROPERTIES ("skip.header.line.count"="1");

**2. PySpark**

**Reading Data from Hive and Performing Analysis:**

from pyspark.sql import SparkSession

# Create a Spark session with Hive support

spark = SparkSession.builder \

.appName("ReadHiveTable") \

.enableHiveSupport() \

.getOrCreate()

# Show all columns

df1 = spark.sql("SELECT \* FROM final\_project.amazon\_stocks")

df1.show()

# Calculate sum of volume per trade date

df2 = spark.sql("SELECT trade\_date, SUM(volume) FROM final\_project.amazon\_stocks GROUP BY trade\_date")

df2.show()

# Calculate the average of volume

df3 = spark.sql("SELECT trade\_date, AVG(volume) FROM final\_project.amazon\_stocks GROUP BY trade\_date")

df3.show()

# Calculate the maximum volume

df4 = spark.sql("SELECT MAX(volume) FROM final\_project.amazon\_stocks")

df4.show()

spark.stop()

**Conclusion**

This project successfully implemented an end-to-end pipeline for processing Amazon stock data. Despite encountering multiple issues, including system crashes and configuration errors, the problems were systematically resolved. The pipeline enables structured data storage, querying, and analysis, demonstrating the integration of NiFi, HDFS, Hive, and PySpark for stock market data processing.