🛠 Big Data Systems Project: Hadoop, Hive, MapReduce & Sqoop Workflows

Tags: *Big Data* | *Hadoop* | *Hive* | *MapReduce* | *Sqoop* | *Streaming* | *TF-IDF* | *Log Processing*

🔍 Project Overview

This end-to-end project involved implementing scalable data processing pipelines using Hadoop and its ecosystem. Across multiple assignments, I explored core components like HDFS, MapReduce, Hadoop Streaming, Hive, and Sqoop to process, clean, aggregate, and analyze real-world datasets ranging from classic text corpora to airline records and system logs. The tasks simulate the real-life complexity of distributed computing and large-scale data engineering.

📦 Summary of Labs / Modules:

📗 Lab 1: Word Count, Cleaning & Vocabulary Richness

Goal: Implement basic MapReduce in Hadoop to compute word frequency, clean data, and compare vocabulary richness.

- Ran MapReduce to count word frequencies in Shakespeare's plays.
- Wrote a custom program to extract the **top 10 most frequent words** from the output.
- Cleaned text by **removing punctuation** and **lowercasing** to standardize words.
- Compared vocabulary richness of Shakespeare vs. Austen using a unique words-to-total words ratio.

📘 Lab 2: Log Processing, Airline Delay Aggregation, and Secondary Sorting

📝 Part 1: Hadoop Log Processing

Goal: Analyze a 10-minute Hadoop log file and count entries by severity per minute.

- Used Hadoop Streaming to compute minute-level counts for **INFO, WARN, ERROR, and FATAL** log entries.
- Built a custom mapper/reducer pipeline and summarized logs in structured format.

🛬 Part 2: Airline Delay Aggregation with Sqoop & MapReduce

Goal: Import relational airline data and compute delay stats.

- Imported relevant columns from a SQL database using **Sqoop** into HDFS.
- Computed **min, max, and average delay per airline carrier** using MapReduce.
- Output sorted by average delay to rank performance of carriers.

🔃 Part 3: Secondary Sorting in Hadoop

Goal: Sort terms by frequency (descending) using only MapReduce (no post-processing).

- Modified the default WordCount logic to emit counts as keys and implemented **secondary sort** logic using Hadoop's sorting guarantees.
- Output: Most frequent words in descending order directly from MapReduce.

📕 Lab 3: Hive for Airline Analysis + TF-IDF Computation

✈️ Part 1: Worst Average Arrival Delay by Airline (Hive)

Goal: Identify airlines with the worst on-time performance.

- Used Hive to compute **average arrival delay per airline** from a preloaded airline dataset.
- Extracted airline names by cleaning the Description column and joined tables on airline ID.
- Exported and compiled results into a clean .txt file for presentation.

📚 Part 2: TF-IDF Pipeline for Document Ranking (Streaming + Hive)

Goal: Build a mini search engine using Hadoop Streaming and Hive.

- Step-by-step MapReduce pipeline to compute **TF-IDF scores**:
  - **Step 1**: Count term frequency per document
  - **Step 2**: Compute total words per document
  - **Step 3**: Split composite keys
  - **Step 4**: Calculate document frequency per term
  - **Step 5**: Combine all data in Hive to compute final TF-IDF
- Final result: A Hive-generated table with (doc_id, term, tfidf_score × 1,000,000)

🔧 Tools & Technologies

- **Hadoop Ecosystem**: HDFS, MapReduce, Hadoop Streaming
- **Data Processing**: Sqoop, Hive
- **Languages**: Python, Shell Scripts, HiveQL
- **Concepts**: TF-IDF, log aggregation, text cleaning, vocabulary richness, airline performance

🧠 Key Takeaways

- Gained practical experience in **distributed data processing and streaming transformations**.
- Learned to optimize **MapReduce logic for custom sorting, aggregation, and multi-step workflows**.
- Demonstrated the use of **Hive for SQL-like processing** and **Sqoop for SQL-HDFS integration**.
- Built a lightweight text ranking pipeline from scratch using **TF-IDF computation**.