# Big Data Pipeline: Hive and Hadoop Implementation

## 1. Introduction

This document outlines the implementation of a data pipeline using Hadoop and Hive, designed to process streaming-like user activity logs and content metadata. The pipeline follows a star schema with a central fact table and supporting dimension tables to enable efficient analytical queries.

## 2. Design Choices

### Data Storage & Partitioning

- Raw Data Storage: Data is first ingested into HDFS in the `/raw/logs/` and `/raw/metadata/` directories.  
- Hive External Tables: Created for raw logs and metadata to provide flexibility in schema evolution.  
- Partitioning Strategy: Fact table partitioned by (year, month, day) to optimize query performance.  
- Columnar Format (Parquet): Fact and dimension tables stored in Parquet format for compression and efficiency.

## 3. Performance Optimization Considerations

By partitioning tables and using the Tez execution engine instead of MapReduce, we optimized query performance and reduced execution times. Resource allocation in YARN further improved efficiency.

## 4. Execution Time Analysis

|  |  |
| --- | --- |
| Stage | Execution Time |
|  |  |
|  |  |
|  |  |
|  |  |
| Data Ingestion (HDFS) | ~10 sec |
| Raw Table Creation | ~2 sec |
| Data Transformation (ETL) | ~15-20 sec |
| Query Execution (Hive) | ~0.1 - 0.3 sec per query |

## 5. Hive Queries Used

### Data Transformation Queries

INSERT OVERWRITE TABLE dim\_users  
SELECT DISTINCT user\_id, region, device FROM raw\_user\_logs;

### Analytical Queries

SELECT dim\_users.region, COUNT(DISTINCT fact\_user\_actions.user\_id) AS active\_users FROM fact\_user\_actions JOIN dim\_users ON fact\_user\_actions.user\_id = dim\_users.user\_id WHERE fact\_user\_actions.year = 2023 AND fact\_user\_actions.month = 9 GROUP BY dim\_users.region;

## 6. Conclusion

This pipeline successfully demonstrates an efficient Hadoop + Hive-based ETL process for processing user activity logs. Future optimizations may include bucketing for further efficiency and leveraging Apache Spark for faster query execution.