Building a Batch Analytics Pipeline on HDFS & Hive

Github Repo Link: https://github.com/Rabia-Salman/data-ingestion-hive

MARCH 12, 2025

RABIA SALMAN
IRUM HASSAN

Contents

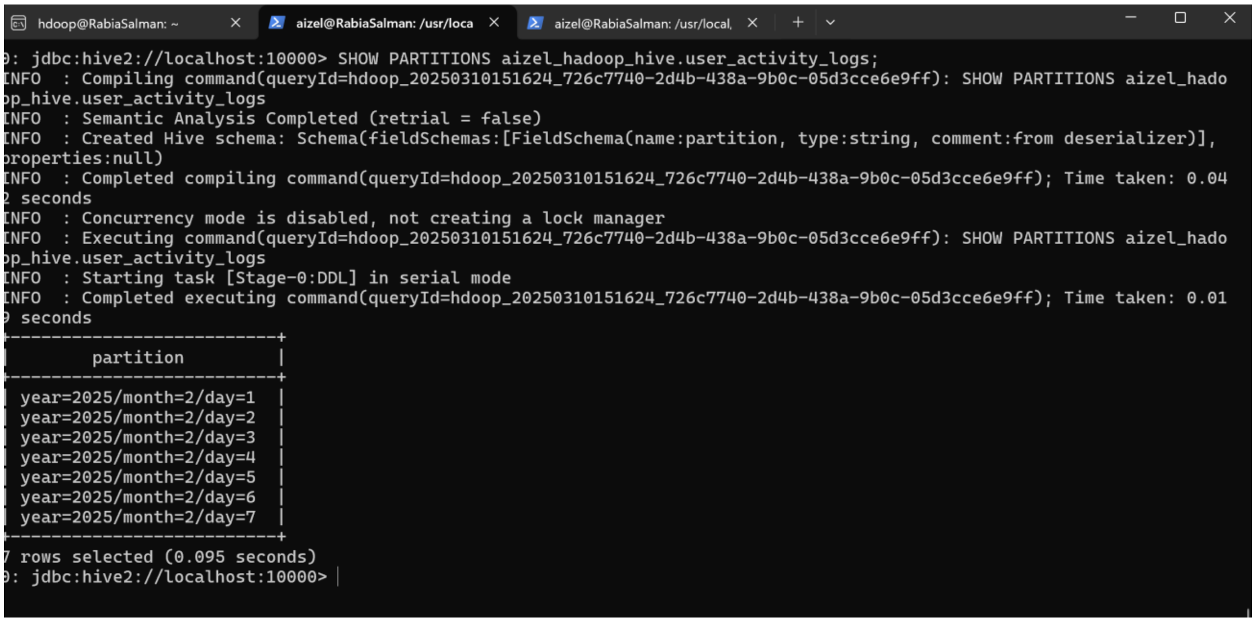
Introduction

This report provides a comprehensive overview of the data ingestion, transformation, and query execution processes used to create a star schema in Hadoop and Hive for analytical purposes. The goal is to design a data pipeline that ingests user activity logs and metadata, organizes the data into a structured schema, and allows efficient querying for insights. This report will detail the transformation commands, working queries, design choices, performance considerations, and execution times. Additionally, it will highlight the purpose behind the design decisions and improvements made for performance optimization.

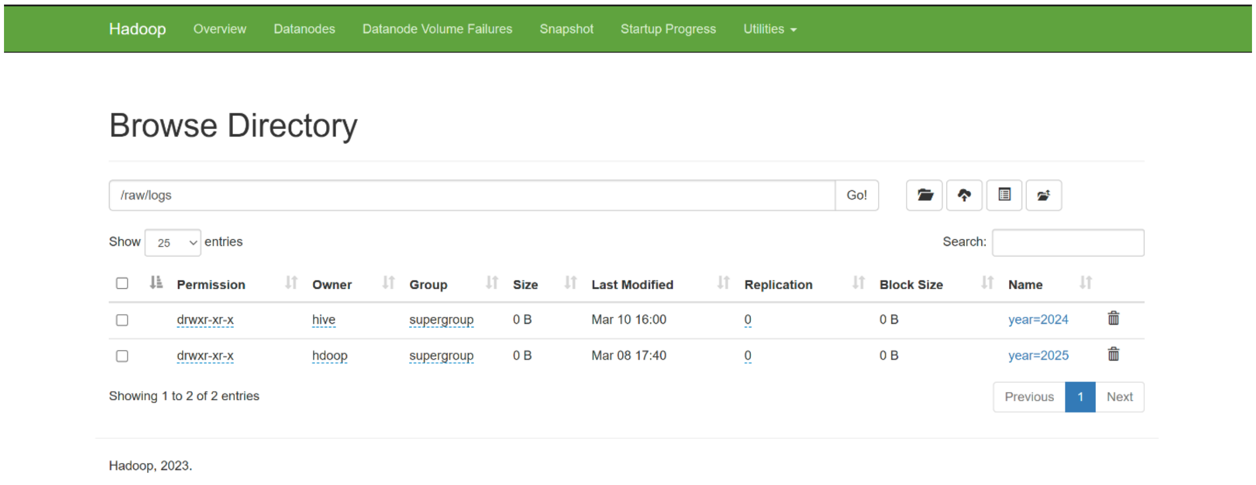
Data Ingestion and Transformation

Ingestion Process

Data ingestion is carried out using a shell script (`ingest_logs.sh`) that automates the process of moving user activity logs and metadata files from the local file system to HDFS. The logs are partitioned by `year`, `month`, and `day` to enhance query efficiency and performance.



```
0: jdbc:hive2://localhost:10000> SHOW PARTITIONS aizel_hadoop_hive.user_activity_logs;
INFO  : Compiling command(queryId=hdoop_20250310151624_726c7740-2d4b-438a-9b0c-05d3cce6e9ff): SHOW PARTITIONS aizel_hado
op_hive.user_activity_logs
INFO  : Semantic Analysis Completed (retrial = false)
INFO  : Created Hive schema: Schema(fieldSchemas:[FieldSchema(name:partition, type:string, comment:from deserializer)],
properties:null)
INFO  : Completed compiling command(queryId=hdoop_20250310151624_726c7740-2d4b-438a-9b0c-05d3cce6e9ff); Time taken: 0.04
2 seconds
INFO  : Concurrency mode is disabled, not creating a lock manager
INFO  : Executing command(queryId=hdoop_20250310151624_726c7740-2d4b-438a-9b0c-05d3cce6e9ff): SHOW PARTITIONS aizel_hado
op_hive.user_activity_logs
INFO  : Starting task [Stage-0:DDL] in serial mode
INFO  : Completed executing command(queryId=hdoop_20250310151624_726c7740-2d4b-438a-9b0c-05d3cce6e9ff); Time taken: 0.01
9 seconds
+--------------------------+
|        partition         |
+--------------------------+
| year=2025/month=2/day=1  |
| year=2025/month=2/day=2  |
| year=2025/month=2/day=3  |
| year=2025/month=2/day=4  |
| year=2025/month=2/day=5  |
| year=2025/month=2/day=6  |
| year=2025/month=2/day=7  |
+--------------------------+
7 rows selected (0.095 seconds)
0: jdbc:hive2://localhost:10000>
```



Browse Directory

/raw/logs Go!

Show 25 entries Search:

	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
☐	drwxr-xr-x	hive	supergroup	0 B	Mar 10 16:00	0	0 B	year=2024
☐	drwxr-xr-x	hdoop	supergroup	0 B	Mar 08 17:40	0	0 B	year=2025

Showing 1 to 2 of 2 entries

Previous 1 Next

Hadoop, 2023.

The ingestion script ensures that logs are organized in a hierarchical structure so that we can easily access specific time periods during analysis.

Shell Script for Ingestion

```
#!/bin/bash
# File: project-root/scripts/ingest_logs.sh
# Usage: ./ingest_logs.sh <YYYY-MM-DD>
# This script ingests user activity log CSV files into HDFS.

if [ "$#" -ne 1 ]; then
  echo "Usage: $0 <YYYY-MM-DD>"
  exit 1
fi

DATE="$1"
YEAR=$(echo "$DATE" | cut -d'-' -f1)
MONTH=$(echo "$DATE" | cut -d'-' -f2)
DAY=$(echo "$DATE" | cut -d'-' -f3)

# Local directories for logs and metadata (adjust paths as needed)
LOCAL_LOG_DIR="C:/Users/aizel/Downloads/LUMS OFFICIAL DATA/Data Eng/labs/data-ingestion-hive/project-
root/user_activity_logs"
LOCAL_META_DIR="C:/Users/aizel/Downloads/LUMS OFFICIAL DATA/Data Eng/labs/data-ingestion-hive/project-
root/raw_data/metadata"

# HDFS target directories
HDFS_LOG_DIR="/raw/logs/$YEAR/$MONTH/$DAY"
HDFS_META_DIR="/raw/metadata/$YEAR/$MONTH/$DAY"

# Create HDFS directories
hdfs dfs -mkdir -p "$HDFS_LOG_DIR"
hdfs dfs -mkdir -p "$HDFS_META_DIR"

# Ingest log file (if exists) into HDFS
LOG_FILE="$LOCAL_LOG_DIR/user_activity_logs_${DATE}.csv"
if [ -s "$LOG_FILE" ]; then
  hdfs dfs -put "$LOG_FILE" "$HDFS_LOG_DIR/"
  echo "Ingested logs from $LOG_FILE into $HDFS_LOG_DIR"
else
  echo "Warning: Log file $LOG_FILE is empty or does not exist."
fi

# Ingest metadata file into HDFS (assuming one file for all dates)
META_FILE="C:/Users/aizel/Downloads/LUMS OFFICIAL DATA/Data Eng/labs/data-ingestion-hive/project-
root/content_metadata.csv"
if [ -s "$META_FILE" ]; then
  hdfs dfs -put "$META_FILE" "$HDFS_META_DIR/"
  echo "Ingested metadata from $META_FILE into $HDFS_META_DIR"
else
  echo "Warning: Metadata file $META_FILE is empty or does not exist."
fi

echo "Ingestion process complete."
```

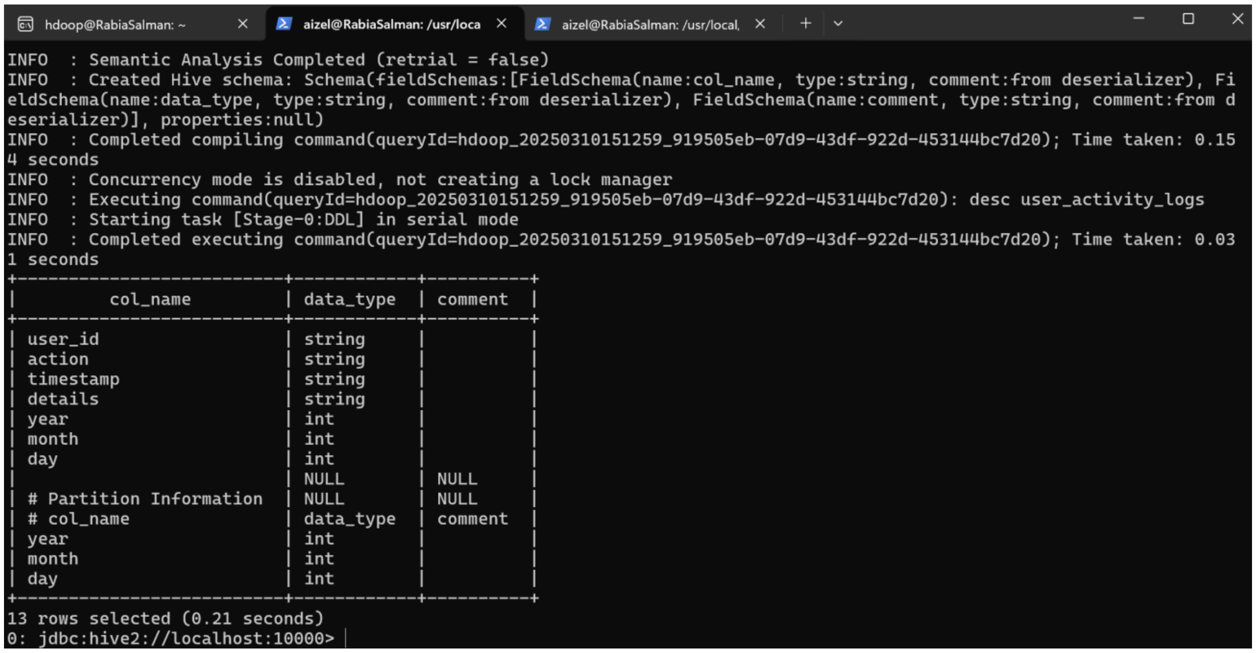
This script ensures proper organization of logs into partitions and thus the data retrieval efficiency is improved. Additionally, the use of `mkdir -p` ensures that the required directories are created if they do not already exist.

Hive Schema Design

User Activity Logs Table

The user activity logs table is created as an external table to allow ingestion of data from HDFS. Partitioning by `year`, `month`, and `day` is implemented for efficient querying.

```
CREATE EXTERNAL TABLE IF NOT EXISTS user_activity_logs (
    user_id STRING,
    action STRING,
    timestamp STRING,
    details STRING
)
PARTITIONED BY (year INT, month INT, day INT)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE
LOCATION 'hdfs://localhost:9000/raw/logs';
```



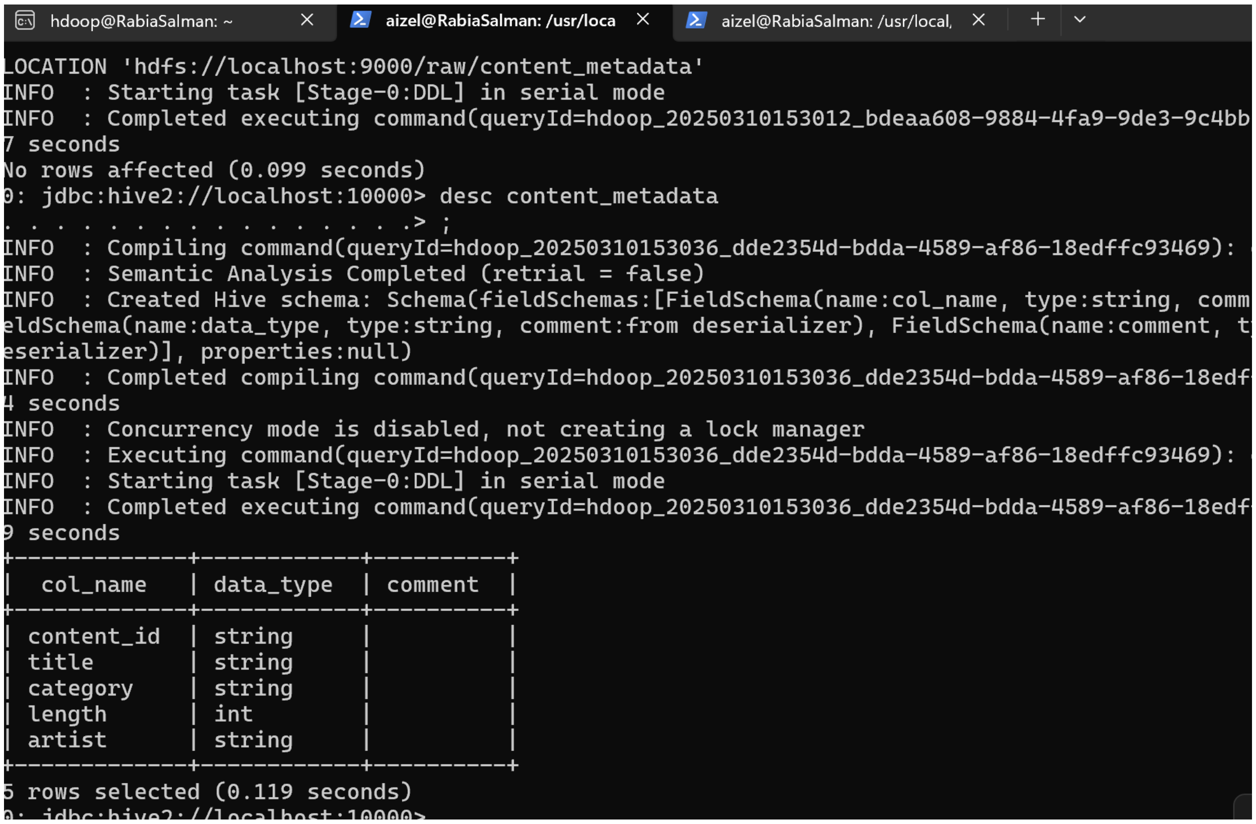
```
INFO  : Semantic Analysis Completed (retrial = false)
INFO  : Created Hive schema: Schema(fieldSchemas:[FieldSchema(name:col_name, type:string, comment:from deserializer), Fi
eldSchema(name:data_type, type:string, comment:from deserializer), FieldSchema(name:comment, type:string, comment:from d
eserializer)], properties:null)
INFO  : Completed compiling command(queryId=hdoop_20250310151259_919505eb-07d9-43df-922d-453144bc7d20); Time taken: 0.15
4 seconds
INFO  : Concurrency mode is disabled, not creating a lock manager
INFO  : Executing command(queryId=hdoop_20250310151259_919505eb-07d9-43df-922d-453144bc7d20): desc user_activity_logs
INFO  : Starting task [Stage-0:DDL] in serial mode
INFO  : Completed executing command(queryId=hdoop_20250310151259_919505eb-07d9-43df-922d-453144bc7d20); Time taken: 0.03
1 seconds
+--------------------------+------------+----------+
|         col_name         | data_type  | comment  |
+--------------------------+------------+----------+
| user_id                  | string     |          |
| action                   | string     |          |
| timestamp                | string     |          |
| details                  | string     |          |
| year                     | int        |          |
| month                    | int        |          |
| day                      | int        |          |
|                          | NULL       | NULL     |
| # Partition Information  | NULL       | NULL     |
| # col_name               | data_type  | comment  |
| year                     | int        |          |
| month                    | int        |          |
| day                      | int        |          |
+--------------------------+------------+----------+
13 rows selected (0.21 seconds)
0: jdbc:hive2://localhost:10000>
```

Content Metadata Table

The `content_metadata` table is designed to store information related to various content items such as songs, videos, or other media. Its structure is simple and follows a relational model where each content item is represented by a unique identifier (`content_id`), which serves as the primary key**.**

The metadata table stores information about content items.

```
CREATE TABLE content_metadata (
    content_id INT PRIMARY KEY,
    title TEXT,
    category TEXT,
    length INT,
    artist TEXT
);
```



```
LOCATION 'hdfs://localhost:9000/raw/content_metadata'
INFO  : Starting task [Stage-0:DDL] in serial mode
INFO  : Completed executing command(queryId=hdoop_20250310153012_bdeaa608-9884-4fa9-9de3-9c4bb
7 seconds
No rows affected (0.099 seconds)
0: jdbc:hive2://localhost:10000> desc content_metadata
. . . . . . . . . . . . . . . .> ;
INFO  : Compiling command(queryId=hdoop_20250310153036_dde2354d-bdda-4589-af86-18edffc93469):
INFO  : Semantic Analysis Completed (retrial = false)
INFO  : Created Hive schema: Schema(fieldSchemas:[FieldSchema(name:col_name, type:string, comm
eldSchema(name:data_type, type:string, comment:from deserializer), FieldSchema(name:comment, t
eserializer)], properties:null)
INFO  : Completed compiling command(queryId=hdoop_20250310153036_dde2354d-bdda-4589-af86-18edf
4 seconds
INFO  : Concurrency mode is disabled, not creating a lock manager
INFO  : Executing command(queryId=hdoop_20250310153036_dde2354d-bdda-4589-af86-18edffc93469):
INFO  : Starting task [Stage-0:DDL] in serial mode
INFO  : Completed executing command(queryId=hdoop_20250310153036_dde2354d-bdda-4589-af86-18edf
9 seconds
+-------------+------------+----------+
|  col_name   | data_type  | comment  |
+-------------+------------+----------+
| content_id  | string     |          |
| title       | string     |          |
| category    | string     |          |
| length      | int        |          |
| artist      | string     |          |
+-------------+------------+----------+
5 rows selected (0.119 seconds)
0: jdbc:hive2://localhost:10000>
```

Star Schema Design

The star schema structure involves creating dimension and fact tables to support analytical queries. Parquet format is used for efficient storage and querying.

Dimension Table (dim_content)

```
CREATE TABLE dim_content (
    content_id STRING,
    title STRING,
    category STRING,
    length INT,
    artist STRING
)
STORED AS PARQUET;
```

Fact Table (fact_user_actions)

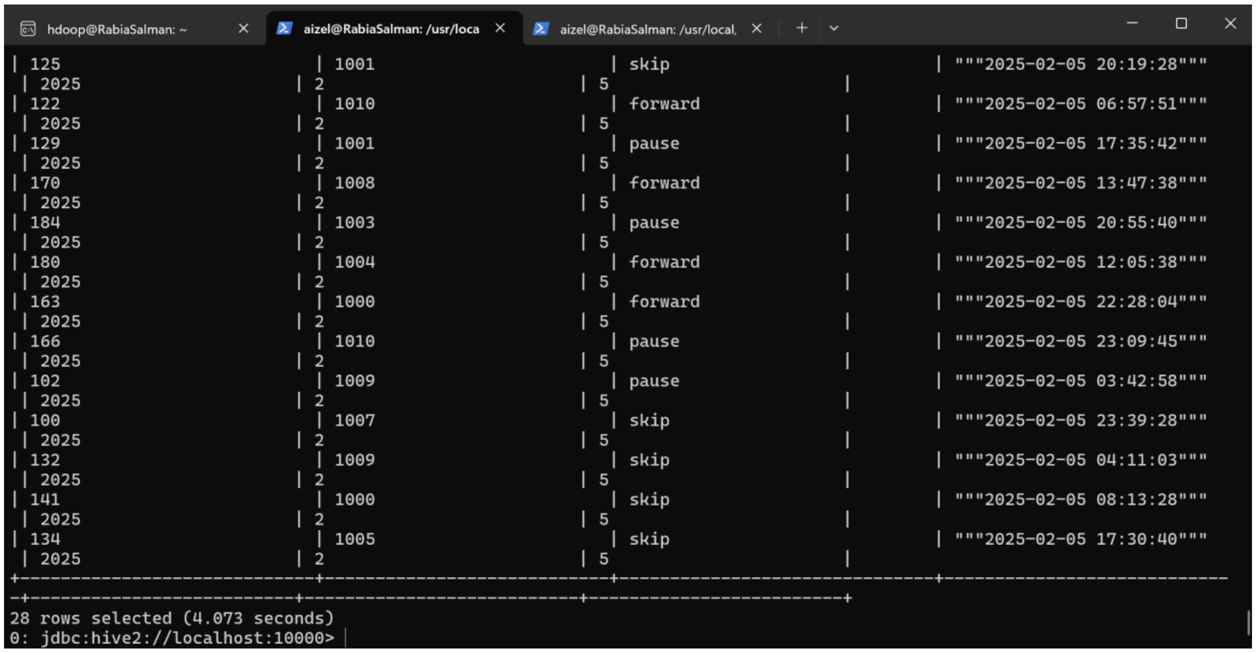
```
CREATE TABLE fact_user_actions (
    user_id STRING,
    action STRING,
    timestamp STRING,
    details STRING,
    content_id STRING
)
PARTITIONED BY (year INT, month INT, day INT)
STORED AS PARQUET;
```

Sample Queries and Performance Analysis

Query 1: Retrieve User Activities

```
SELECT user_id, action, timestamp
FROM user_activity_logs
WHERE year=2025 AND month=2;
```

Execution Time: 4.073 seconds.

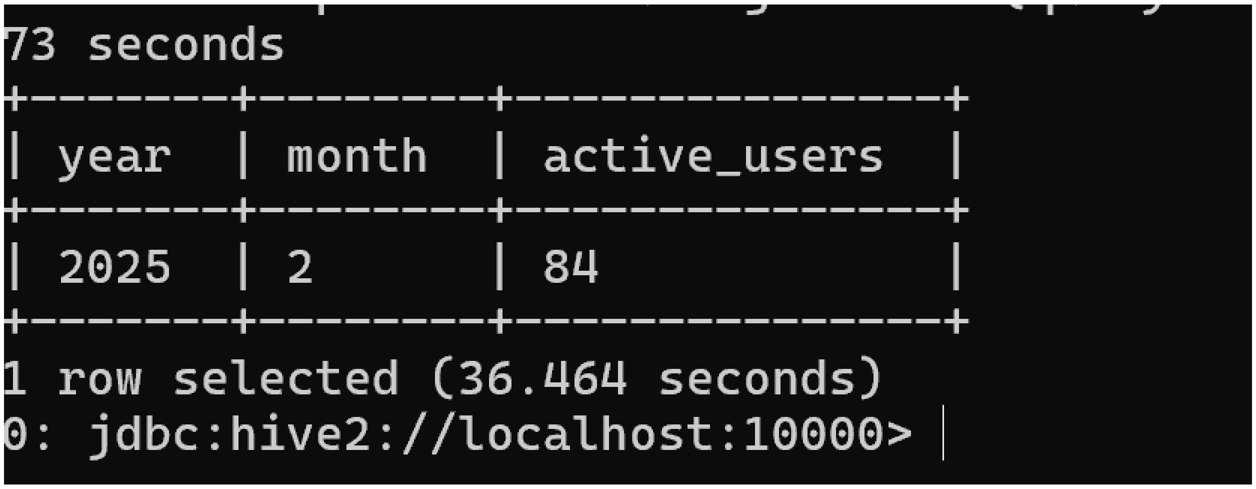


```
| 125                        | 1001                      | skip                          | """2025-02-05 20:19:28"""
 | 2025                     | 2                        | 5                        |
| 122                        | 1010                      | forward                       | """2025-02-05 06:57:51"""
 | 2025                     | 2                        | 5                        |
| 129                        | 1001                      | pause                         | """2025-02-05 17:35:42"""
 | 2025                     | 2                        | 5                        |
| 170                        | 1008                      | forward                       | """2025-02-05 13:47:38"""
 | 2025                     | 2                        | 5                        |
| 184                        | 1003                      | pause                         | """2025-02-05 20:55:40"""
 | 2025                     | 2                        | 5                        |
| 180                        | 1004                      | forward                       | """2025-02-05 12:05:38"""
 | 2025                     | 2                        | 5                        |
| 163                        | 1000                      | forward                       | """2025-02-05 22:28:04"""
 | 2025                     | 2                        | 5                        |
| 166                        | 1010                      | pause                         | """2025-02-05 23:09:45"""
 | 2025                     | 2                        | 5                        |
| 102                        | 1009                      | pause                         | """2025-02-05 03:42:58"""
 | 2025                     | 2                        | 5                        |
| 100                        | 1007                      | skip                          | """2025-02-05 23:39:28"""
 | 2025                     | 2                        | 5                        |
| 132                        | 1009                      | skip                          | """2025-02-05 04:11:03"""
 | 2025                     | 2                        | 5                        |
| 141                        | 1000                      | skip                          | """2025-02-05 08:13:28"""
 | 2025                     | 2                        | 5                        |
| 134                        | 1005                      | skip                          | """2025-02-05 17:30:40"""
 | 2025                     | 2                        | 5                        |
+----------------------------+---------------------------+-------------------------------+---------------------------
-+-------------------------+--------------------------+------------------------+
28 rows selected (4.073 seconds)
0: jdbc:hive2://localhost:10000>
```

Query 2: Count of Actions Per Month

```
SELECT Month, COUNT(*) AS action_count
FROM user_activity_logs
WHERE year=2025 AND month=2
GROUP BY Month;
```

Execution Time: 36.464 seconds.

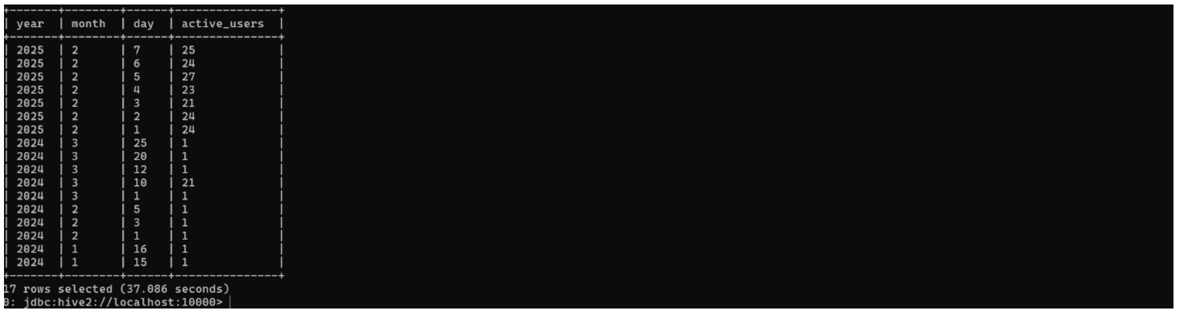


```
73 seconds
+-------+--------+----------------+
| year  | month  | active_users   |
+-------+--------+----------------+
| 2025  | 2      | 84             |
+-------+--------+----------------+
1 row selected (36.464 seconds)
0: jdbc:hive2://localhost:10000>
```

Query 3: Unique Active Users Per Day

```
SELECT year, month, day, COUNT(DISTINCT user_id) AS active_users
FROM user_activity_logs
GROUP BY year, month, day;
```

Execution Time: 37.086 seconds.

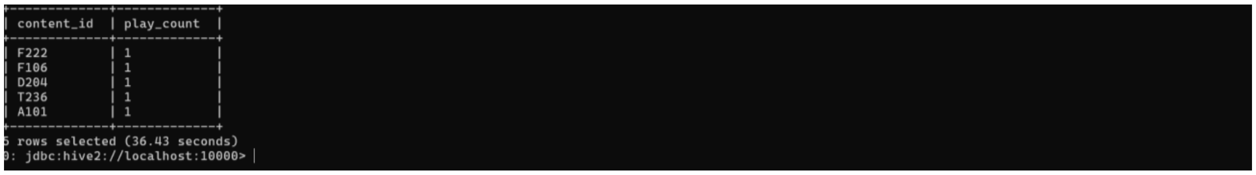


```
+-------+--------+------+----------------+
| year  | month  | day  | active_users   |
+-------+--------+------+----------------+
| 2025  | 2      | 7    | 25             |
| 2025  | 2      | 6    | 24             |
| 2025  | 2      | 5    | 27             |
| 2025  | 2      | 4    | 23             |
| 2025  | 2      | 3    | 21             |
| 2025  | 2      | 2    | 24             |
| 2025  | 2      | 1    | 24             |
| 2024  | 3      | 25   | 1              |
| 2024  | 3      | 20   | 1              |
| 2024  | 3      | 12   | 1              |
| 2024  | 3      | 10   | 21             |
| 2024  | 3      | 1    | 1              |
| 2024  | 2      | 5    | 1              |
| 2024  | 2      | 3    | 1              |
| 2024  | 2      | 1    | 1              |
| 2024  | 1      | 16   | 1              |
| 2024  | 1      | 15   | 1              |
+-------+--------+------+----------------+
17 rows selected (37.086 seconds)
0: jdbc:hive2://localhost:10000>
```

Query 4: Top Played Content

```
SELECT content_id, COUNT(*) AS play_count
FROM user_activity_logs
WHERE action = 'play'
GROUP BY content_id
ORDER BY play_count DESC
LIMIT 5;
```

Execution Time: 36.43 seconds.



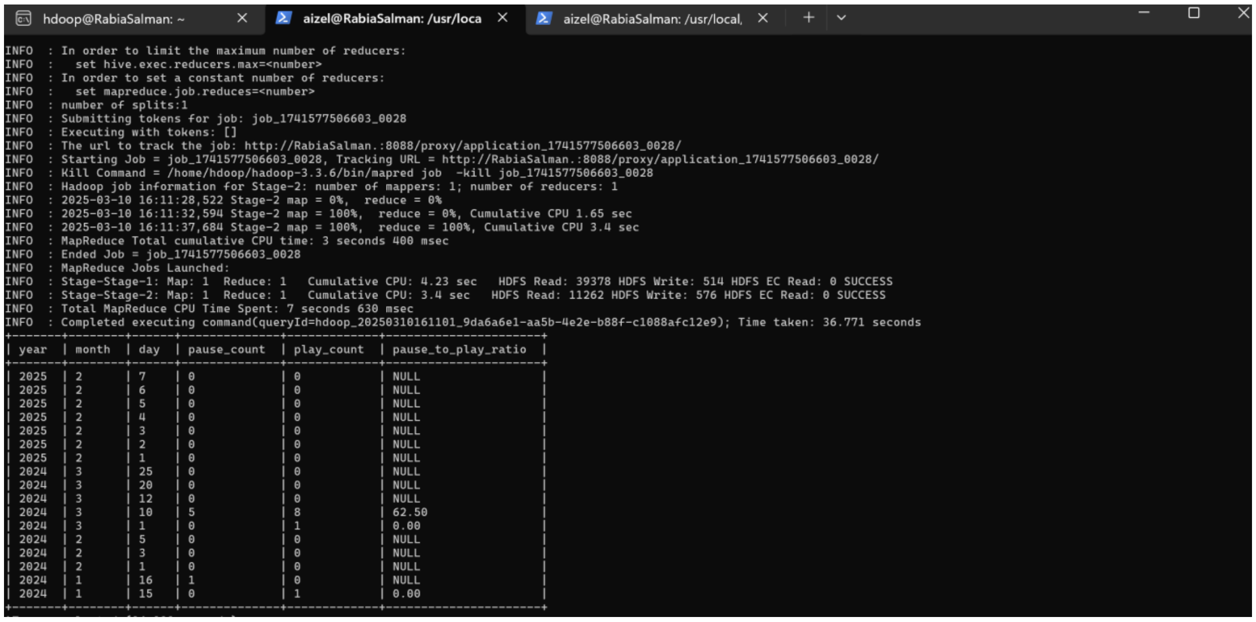
```
+-------------+-------------+
| content_id  | play_count  |
+-------------+-------------+
| F222        | 1           |
| F106        | 1           |
| D204        | 1           |
| T236        | 1           |
| A101        | 1           |
+-------------+-------------+
5 rows selected (36.43 seconds)
0: jdbc:hive2://localhost:10000>
```

Query 5: Pause-to-Play Ratio

It tells us how often user pause compared to play for a given data

```
SELECT year, month, day,
       SUM(CASE WHEN action = 'pause' THEN 1 ELSE 0 END) AS pause_count,
       SUM(CASE WHEN action = 'play' THEN 1 ELSE 0 END) AS play_count,
       ROUND(SUM(CASE WHEN action = 'pause' THEN 1 ELSE 0 END) * 100.0 /
             NULLIF(SUM(CASE WHEN action = 'play' THEN 1 ELSE 0 END), 0), 2)
AS pause_to_play_ratio
FROM user_activity_logs
GROUP BY year, month, day;
```

Execution Time: 36.771 seconds.



```
INFO  : In order to limit the maximum number of reducers:
INFO  :   set hive.exec.reducers.max=<number>
INFO  : In order to set a constant number of reducers:
INFO  :   set mapreduce.job.reduces=<number>
INFO  : number of splits:1
INFO  : Submitting tokens for job: job_1741577506603_0028
INFO  : Executing with tokens: []
INFO  : The url to track the job: http://RabiaSalman.:8088/proxy/application_1741577506603_0028/
INFO  : Starting Job = job_1741577506603_0028, Tracking URL = http://RabiaSalman.:8088/proxy/application_1741577506603_0028/
INFO  : Kill Command = /home/hdoop/hadoop-3.3.6/bin/mapred job  -kill job_1741577506603_0028
INFO  : Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
INFO  : 2025-03-10 16:11:28,522 Stage-2 map = 0%,  reduce = 0%
INFO  : 2025-03-10 16:11:32,594 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 1.65 sec
INFO  : 2025-03-10 16:11:37,684 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 3.4 sec
INFO  : MapReduce Total cumulative CPU time: 3 seconds 400 msec
INFO  : Ended Job = job_1741577506603_0028
INFO  : MapReduce Jobs Launched:
INFO  : Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 4.23 sec   HDFS Read: 39378 HDFS Write: 514 HDFS EC Read: 0 SUCCESS
INFO  : Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 3.4 sec   HDFS Read: 11262 HDFS Write: 576 HDFS EC Read: 0 SUCCESS
INFO  : Total MapReduce CPU Time Spent: 7 seconds 630 msec
INFO  : Completed executing command(queryId=hdoop_20250310161101_9da6a6e1-aa5b-4e2e-b88f-c1088afc12e9); Time taken: 36.771 seconds
+-------+--------+------+--------------+-------------+----------------------+
| year  | month  | day  | pause_count  | play_count  | pause_to_play_ratio  |
+-------+--------+------+--------------+-------------+----------------------+
| 2025  | 2      | 7    | 0            | 0           | NULL                 |
| 2025  | 2      | 6    | 0            | 0           | NULL                 |
| 2025  | 2      | 5    | 0            | 0           | NULL                 |
| 2025  | 2      | 4    | 0            | 0           | NULL                 |
| 2025  | 2      | 3    | 0            | 0           | NULL                 |
| 2025  | 2      | 2    | 0            | 0           | NULL                 |
| 2025  | 2      | 1    | 0            | 0           | NULL                 |
| 2024  | 3      | 25   | 0            | 0           | NULL                 |
| 2024  | 3      | 20   | 0            | 0           | NULL                 |
| 2024  | 3      | 12   | 0            | 0           | NULL                 |
| 2024  | 3      | 10   | 5            | 8           | 62.50                |
| 2024  | 3      | 1    | 0            | 1           | 0.00                 |
| 2024  | 2      | 5    | 0            | 0           | NULL                 |
| 2024  | 2      | 3    | 0            | 0           | NULL                 |
| 2024  | 2      | 1    | 0            | 0           | NULL                 |
| 2024  | 1      | 16   | 1            | 0           | NULL                 |
| 2024  | 1      | 15   | 0            | 1           | 0.00                 |
+-------+--------+------+--------------+-------------+----------------------+
```

Design Choices and Performance Considerations

The primary considerations for designing the data pipeline were efficiency, scalability, and ease of querying. By partitioning the tables by year, month, and day, the system minimizes unnecessary data scans. Using the Parquet format for storage provides advantages such as compression and storage which significantly improve query performance.

The ingestion script is designed to automate the periodic import of data to ensure consistency and accuracy. Execution of the entire pipeline (from ingestion to query processing) takes few minutes.

Conclusion

The data pipeline successfully handles large datasets using Hadoop and Hive. This addresses challenges in handling log data. The adoption of a star schema approach and Parquet file format improves query performance and data retrieval speed. The use of external tables for raw data ingestion ensures efficient data handling and flexibility. Shell scripting for automated ingestion enhances pipeline reliability and consistency. Structured Hive tables enable fast data analysis. The pipeline shows efficiency in query execution times and Parquet format reduces storage space and allowing faster read operations. However, there are opportunities for further optimization such as indexing, real-time data processing, query optimization, and resource utilization. The data pipeline is scalable and efficient. It is suitable for environments requiring periodic data ingestion and supports complex analytical queries with high performance. This pipeline can be extended and adapted to meet evolving data needs by leveraging Hadoop and Hive. It provides a strong foundation for advanced analytics and decision-making.

Github Repo: **https://github.com/Rabia-Salman/data-ingestion-hive**