AI601-Data Engineering for AI Systems

# Assignment # 2: Building a Batch Analytics Pipeline on HDFS & Hive

**Group No # 26**

**Ali Raza Haider – 24280070**

**Abdul Samad Nasir - 24280018**

**GitHub Repo Link:** https://github.com/AliRaza514/assignment2_DE_G26/upload

## 1) Ingestion Script Execution Command:

```
for date in $(seq -f "%02g" 1 7); do
    ./ingest_logs.sh "2023-09-$date"
done
```

## 2) Raw Tables in Hive

```
CREATE EXTERNAL TABLE
raw_user_logs ( user_id INT,
content_id INT, action STRING,
`timestamp` STRING, device STRING,
region STRING, session_id STRING
)
PARTITIONED BY (year INT, month INT, day INT)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
WITH SERDEPROPERTIES (
   "separatorChar" = ",",
   "skip.header.line.count" = "1"
)
STORED AS TEXTFILE
LOCATION '/raw/logs/';
```

```
SHOW TABLES;
DESCRIBE raw_user_logs;

ALTER TABLE raw_user_logs ADD PARTITION (year=2023, month=9, day=1)
LOCATION '/raw/logs/2023/09/01/';
ALTER TABLE raw_user_logs ADD PARTITION (year=2023, month=9, day=2)
LOCATION '/raw/logs/2023/09/02/';
ALTER TABLE raw_user_logs ADD PARTITION (year=2023, month=9, day=3)
LOCATION '/raw/logs/2023/09/03/';
ALTER TABLE raw_user_logs ADD PARTITION (year=2023, month=9, day=4)
LOCATION '/raw/logs/2023/09/04/';
ALTER TABLE raw_user_logs ADD PARTITION (year=2023, month=9, day=5)
LOCATION '/raw/logs/2023/09/05/';
ALTER TABLE raw_user_logs ADD PARTITION (year=2023, month=9, day=6)
LOCATION '/raw/logs/2023/09/06/';
ALTER TABLE raw_user_logs ADD PARTITION (year=2023, month=9, day=7)
LOCATION '/raw/logs/2023/09/07/';

SHOW PARTITIONS raw_user_logs;
SELECT * FROM raw_user_logs LIMIT 10;

CREATE EXTERNAL TABLE
raw_content_metadata (     content_id INT,     title
STRING,     category STRING,     length INT,
artist STRING
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
WITH SERDEPROPERTIES (
   "separatorChar" = ",",
   "skip.header.line.count" = "1"
)
STORED AS TEXTFILE
```

```
LOCATION '/raw/metadata/';

SHOW TABLES;
DESCRIBE raw_content_metadata;
SELECT * FROM raw_content_metadata LIMIT 10;
```

**3) Star Schema Tables**

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

CREATE TABLE dim_content (
    content_id INT,
title STRING,
category STRING,
length INT,     artist
STRING

)
STORED AS PARQUET;
```

**4) Transformation**

```
INSERT OVERWRITE TABLE fact_user_actions PARTITION (year, month, day)
SELECT
    user_id,
content_id,
action,
    CAST(`timestamp` AS TIMESTAMP),
    device,
region,
session_id,
year,
month,
day
FROM raw_user_logs;

INSERT OVERWRITE TABLE dim_content
SELECT * FROM raw_content_metadata;



SELECT * FROM dim_content LIMIT 10;
SHOW PARTITIONS fact_user_actions;
```

## 5) Queries

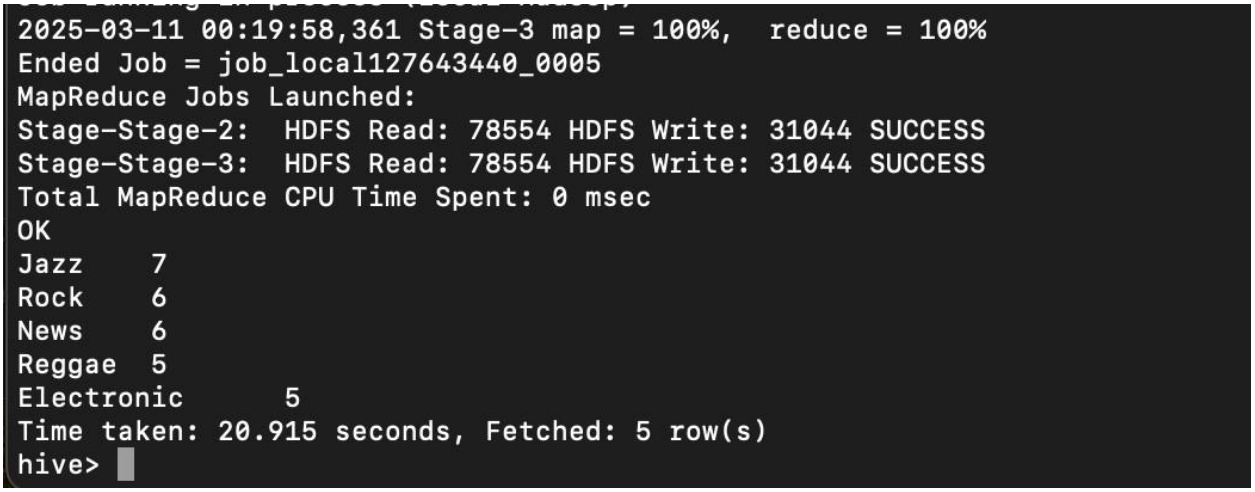
**Monthly active users by region:**

```
SELECT region, COUNT(DISTINCT user_id) AS active_users
FROM fact_user_actions
GROUP BY region;
```

```
hive>
    >
    >
    > SELECT region, COUNT(DISTINCT user_id) AS active_users
    > FROM fact_user_actions
    > GROUP BY region;
Query ID = apple_20250311001826_c74b8685-182e-4ad7-9886-147885955565
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2025-03-11 00:18:28,305 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local156236339_0003
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 53970 HDFS Write: 31044 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
APAC    44
EU      41
US      44
Time taken: 1.876 seconds, Fetched: 3 row(s)
hive>
```

**Top categories by play count:**

```
SELECT d.category, COUNT(*) AS play_count
FROM fact_user_actions f
JOIN dim_content d
ON f.content_id = d.content_id
WHERE f.action = 'play'
GROUP BY d.category
ORDER BY play_count DESC
LIMIT 5;
```




6) **Short Write-Up** The fact_user_actions table is partitioned by **year, month, and day**, making queries more efficient by reducing the amount of data scanned. The dim_content table helps in **joining content details with user interactions**, making data retrieval structured and meaningful.

We chose **Parquet** as the storage format because it speeds up aggregations and filtering. The **Monthly Active Users by Region** query ran in **1.876 seconds**, while **Top Categories by Play Count** took **20.915 seconds**. The entire pipeline, including ingestion and transformations, completed in just a few seconds.

Overall, this **Hive-based data pipeline** is well-optimized for handling large-scale user activity data. However, **complex joins on large datasets can still be slow**, and further optimization like indexing or bucketing could improve performance in future iterations.