AI601-Data Engineering for AI Systems

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

**Group No: 38**

**Muhammad Annus Shabbir – 24280015**

**Muhammad Atif - 24280067**

**GitHub Repo Link:** [*https://github.com/annus-lums/ai601\_assignment\_2*](https://github.com/annus-lums/ai601_assignment_2)

1. **Ingestion Script Execution Command***:*

*for date in $(seq -f "%02g" 1 7); do*

*./ingest\_logs.sh "2023-09-$date"*

*Done*

1. **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;*

1. **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;*

1. **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;*

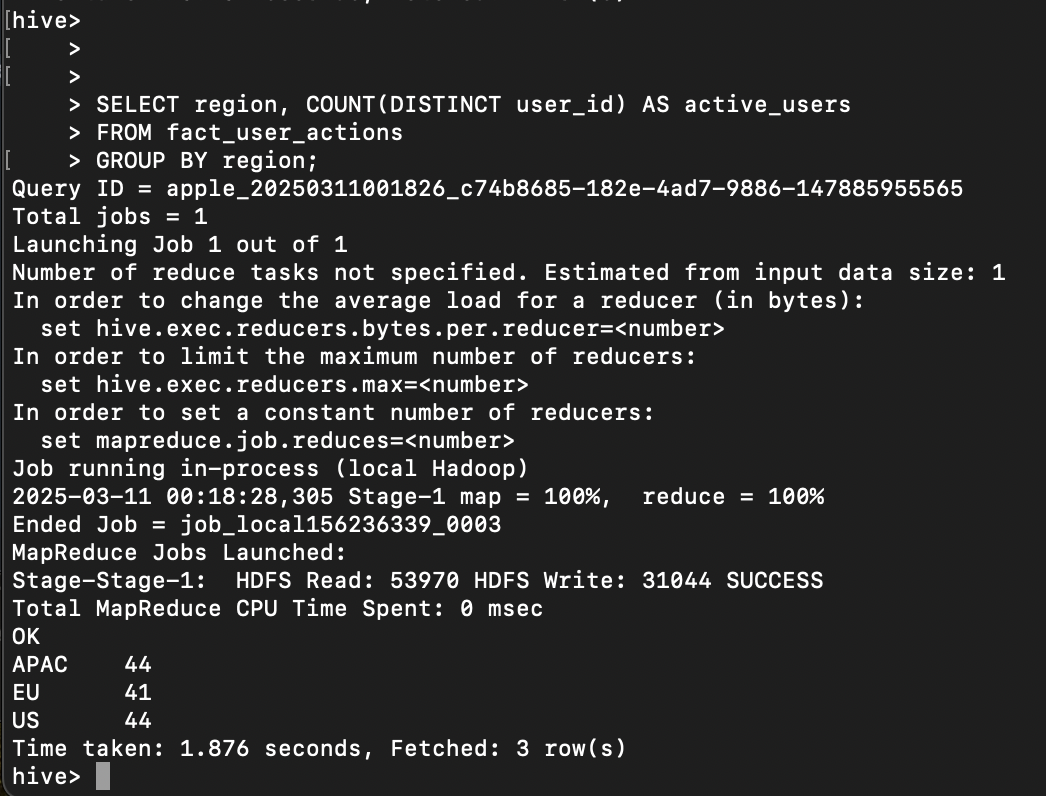
1. **Queries**

***Monthly active users by region:***

*SELECT region, COUNT(DISTINCT user\_id) AS active\_users*

*FROM fact\_user\_actions*

*GROUP BY region;*

**

***Top categories by play count:***

*SELECT d.category, COUNT(\*) AS play\_count*

*FROM fact\_user\_actions f*

*JOIN dim\_content d*

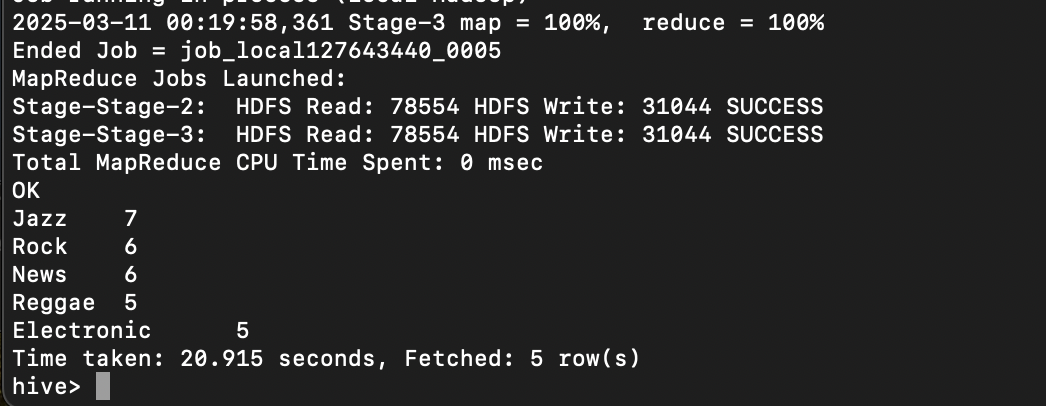
*ON f.content\_id = d.content\_id*

*WHERE f.action = 'play'*

*GROUP BY d.category*

*ORDER BY play\_count DESC*

*LIMIT 5;*

**

1. **Short Write-Up**

* fact\_user\_actions is partitioned by date (year, month, day) to improve query performance
* dim\_content enables joins with the fact table and contains details of content.
* Parquet used for faster for aggregations and filtering
* The Monthly Active Users by Region query took 1.876 seconds
* The Top Categories by Play Count query took 20.915 seconds
* The piepeline execution time of various steps were few seconds.
* This Hive-based data pipeline enables efficient storage and querying of large-scale user interaction data. Parquet and Partitioning significantly improves performance, but complex joins on large tables still take time. Future optimizations can further enhance query speeds.