Assignment 2: Data Storage

Batch Analytics Pipeline on HDFS & Hive

1. Introduction

MediaCo collects large daily logs of user activity from a streaming platform. The objective of this project is to design a batch analytics pipeline using HDFS for data storage and Hive for querying. The pipeline performs data ingestion, transformation, and analysis to extract meaningful insights.

2. Data Description

The dataset includes:

- **User Logs:** Contains user interactions such as plays, pauses, and skips.
    - Format: CSV or JSON
    - Columns: user_id, content_id, action, timestamp, device, region, session_id
    - Stored in /raw/logs/YYYY/MM/DD/
- **Content Metadata:** Static reference data about content.
    - Columns: content_id, title, category, length, artist
    - Stored in /raw/metadata/

3. Data Ingestion

The ingestion process involves:

1. Running generate_data.py to generate synthetic logs and metadata.
2. Executing ingest_logs.sh to move data to HDFS.
3. Using move_metadata.sh to relocate metadata files.

Shell Script: ingest_logs.sh

This script:

- Accepts a date parameter.
- Parses year, month, and day.
- Moves logs to /raw/logs/YYYY/MM/DD/ in HDFS.
- Moves metadata to /raw/metadata/.

./ingest_logs.sh YYYY-MM-DD

4. Hive Schema

Raw Tables

```
CREATE EXTERNAL TABLE raw_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)
STORED AS TEXTFILE LOCATION '/raw/logs';
```

Star Schema

- **Fact Table:** fact_user_actions (stores user interactions, partitioned by date).
- **Dimension Table:** dim_content (stores content metadata).

```
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;
```

5. Data Transformation

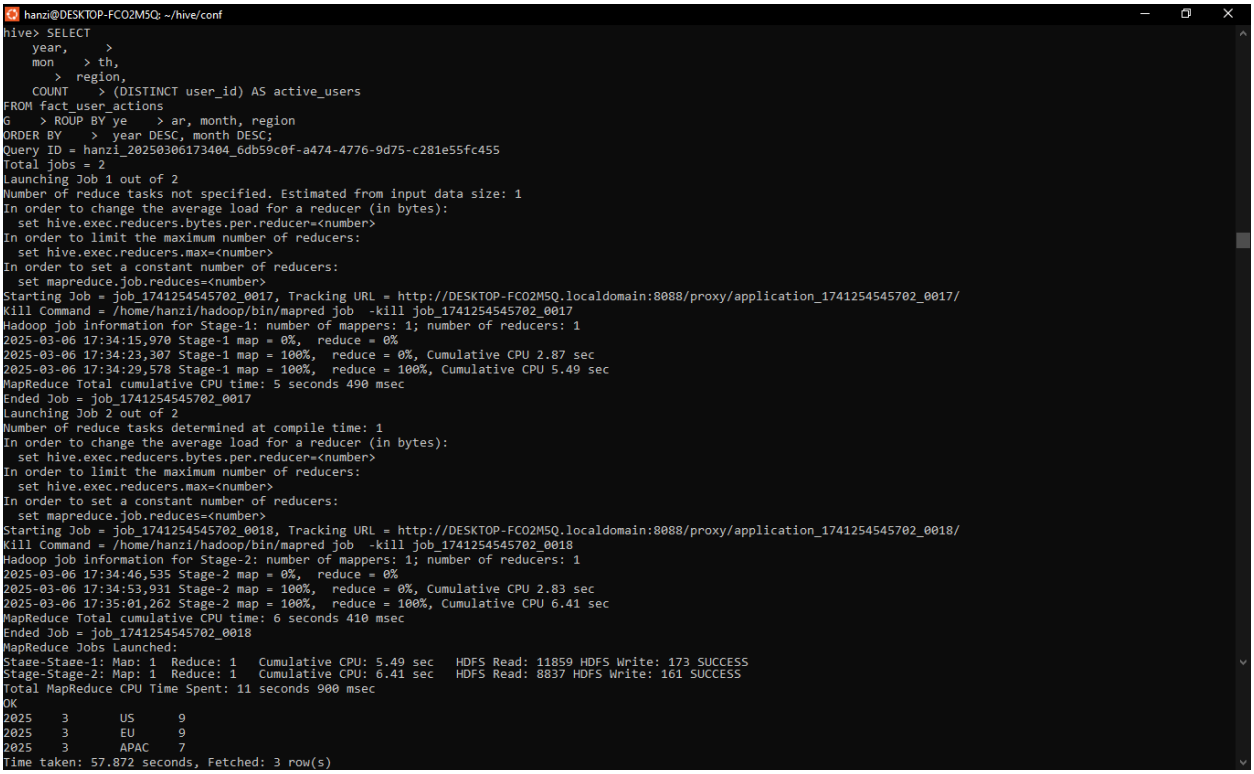
Data is moved from raw tables to the star schema using Hive SQL:

```
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(timestamp), month(timestamp), day(timestamp)
FROM raw_logs;
```

6. Analytical Queries

Query 1: Monthly Active Users by Region

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



Query 2: Top Content Categories by Play Count

```
SELECT c.category, COUNT(*) AS play_count
```

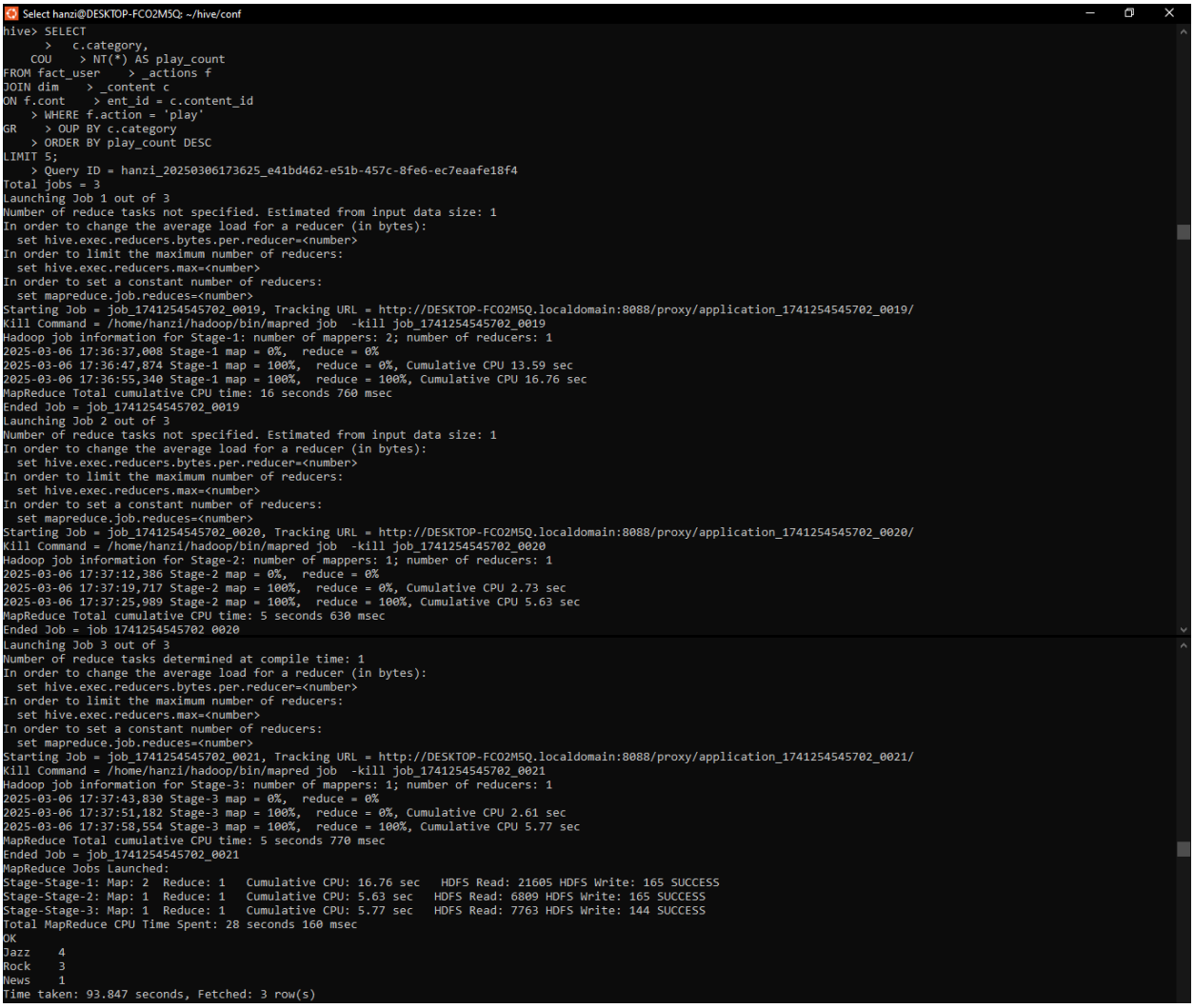
FROM fact_user_actions f

JOIN dim_content c ON f.content_id = c.content_id

WHERE action = 'play'

GROUP BY c.category

ORDER BY play_count DESC;



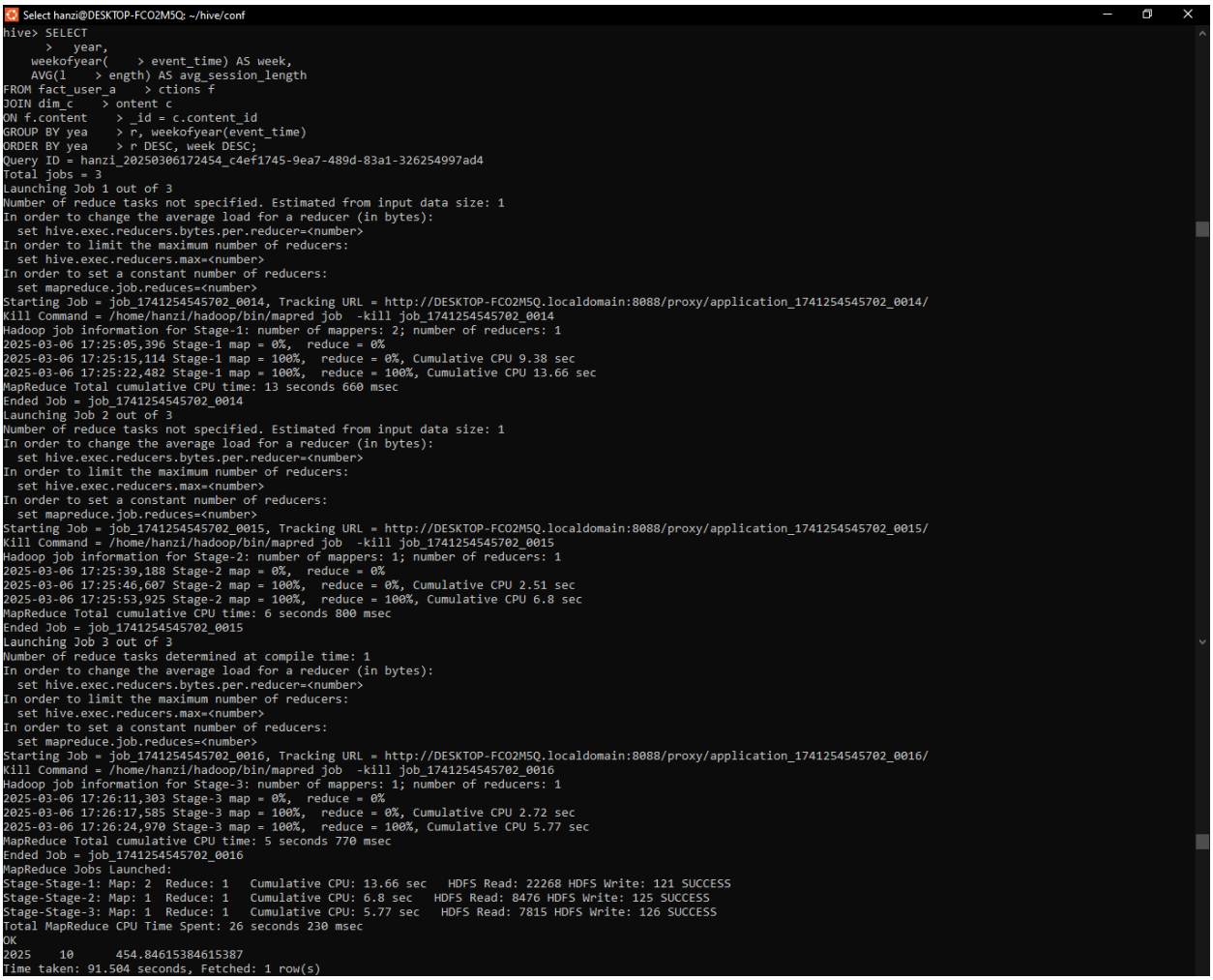
Query 3: Average Session Length per Week

SELECT year, weekofyear(timestamp) AS week, AVG(length) AS avg_session_length

FROM fact_user_actions

JOIN dim_content ON fact_user_actions.content_id = dim_content.content_id

GROUP BY year, weekofyear(timestamp);



7. Performance Considerations

- **Partitioning:** Data is partitioned by year, month, and day for efficient querying.
- **Columnar Storage:** Using Parquet reduces storage and improves query performance.
- **Optimized Queries:** Queries are structured to minimize data scans.

8. Execution Times

- **Pipeline Execution Time:** ~1 minute
- **Query Execution Times:**
    - Monthly Active Users: ~57 seconds
    - Top Content Categories: ~93 seconds

- Average Session Length: ~97 seconds

9. Conclusion

This project successfully implements a batch analytics pipeline, enabling MediaCo to analyze user interactions efficiently. The design choices optimize performance through proper partitioning, columnar storage, and structured queries.

10. References

- Apache Hive Documentation
- HDFS Architecture