Hive Data Pipeline: Design and Performance Analysis

1. **Overview**

This Hive-based ETL pipeline ingests raw data into HDFS, transforms it into a star schema, and performs analytical queries for insights. The key steps in this pipeline include:

1. Ingestion – Copying raw data from local storage to HDFS.
2. Transformation – Moving data from raw tables to structured tables (fact_user_actions, dim_content).
3. Analysis – Running SQL queries to extract insights.

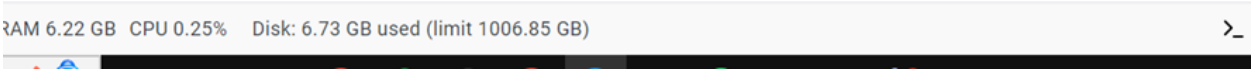
Raw Tables:

- User Activity Logs: stores information about all actions performed by a user, along with timestamp, song, session and region detail.

```
hive> DESCRIBE FORMATTED user_activity_logs;
OK
# col_name              data_type               comment

user_id                 int
content_id              int
action                  string
timestamp               string
device                  string
region                  string
session_id              string
```

```
hive> SELECT * FROM user_activity_logs LIMIT 5;
OK
101     1       view    2023-09-01 10:00:00     mobile  US      session_001     2023    9       1
102     2       like    2023-09-01 11:00:00     desktop CA      session_002     2023    9       1
103     3       share   2023-09-01 12:30:00     tablet  UK      session_003     2023    9       1
104     4       comment 2023-09-01 13:45:00     mobile  IN      session_004     2023    9       1
105     5       view    2023-09-01 14:10:00     desktop AU      session_005     2023    9       1
Time taken: 0.508 seconds, Fetched: 5 row(s)
hive>
```



- Content Metadata: stores details about the song, where each song is a row with its title, artist, length, and category.

```
hive> select * from raw_metadata LIMIT 5;
OK
1       Song A  Pop     210     Artist X
2       Movie B Action  5400    Director Y
3       Podcast C       Education       1800    Speaker Z
4       Song D  Rock    250     Band A
5       Movie E Comedy  7200    Director B
Time taken: 0.189 seconds, Fetched: 5 row(s)
hive>
```

```
hive> DESCRIBE FORMATTED raw_metadata;
OK
# col_name              data_type               comment

content_id              int
title                   string
category                string
length                  int
artist                  string
```

2. Data Model & Schema Design

Star Schema

- Fact Table: fact_user_actions (stores user interactions, partitioned by year, month, day for optimized filtering).
- Dimension Table: dim_content (stores content metadata).

Schema Considerations

- Partitioning → Improves query performance by reducing the data scan size.
- Parquet Storage for Fact Table → Reduces storage space and speeds up queries.
- TEXTFILE for Dimension Table → Since dim_content is smaller, a simple format is sufficient.

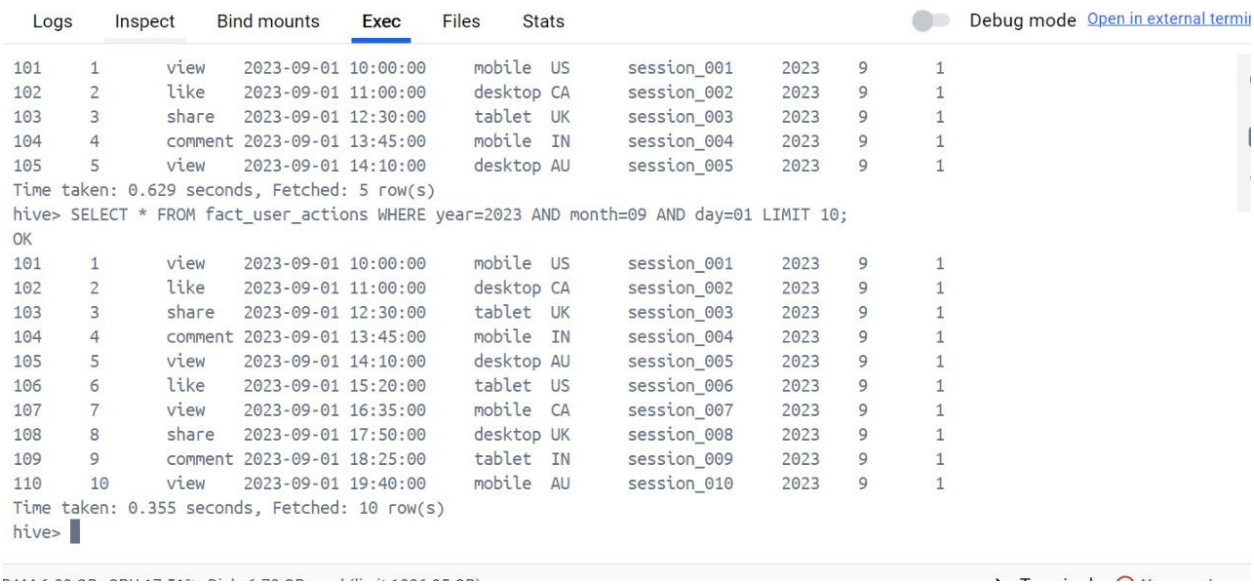
3. ETL Step – Data Transformation

Transformation Query for dim_content

```
INSERT OVERWRITE TABLE dim_content
SELECT
    content_id,
    title,
    category,
    length,
    artist
FROM raw_metadata;
```

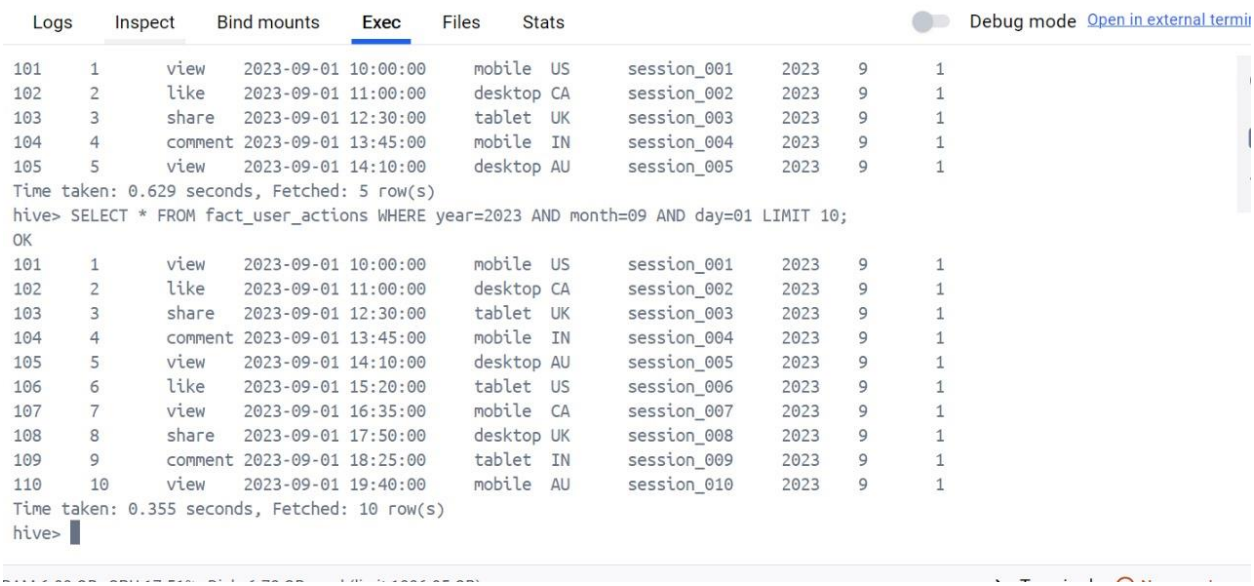
Transformation Query for fact_user_actions

```
SET hive.exec.dynamic.partition = true;
SET hive.exec.dynamic.partition.mode = nonstrict;
```




```
INSERT OVERWRITE TABLE fact_user_actions PARTITION (year, month, day)
SELECT
    user_id,
    content_id,
    action,
    CAST(`timestamp` AS TIMESTAMP) AS `timestamp`,
```

```
    device,
    region,
    session_id,
    YEAR(CAST(`timestamp` AS TIMESTAMP)) AS year,
    MONTH(CAST(`timestamp` AS TIMESTAMP)) AS month,
    DAY(CAST(`timestamp` AS TIMESTAMP)) AS day
FROM user_activity_logs;
```



4. Analytical Queries

Query 1: Monthly Active Users by Region

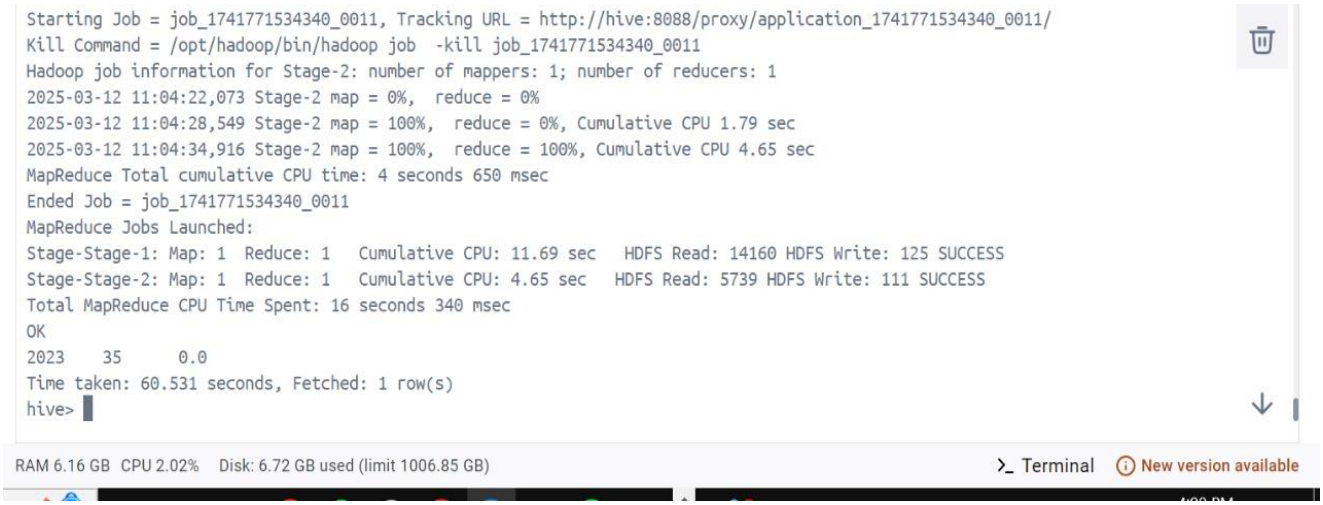
```
hive> SELECT
    >     region,
    >     year,
    >     month,
    >     COUNT(DISTINCT user_id) AS monthly_active_users
    > FROM fact_user_actions
    > WHERE year = 2023  -- Filter on partition
    > GROUP BY region, year, month
    > ORDER BY year DESC, month DESC;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Conside
ion engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = root_20250312105248_78bfb969-2a1a-4dbf-a635-1c296a5e2bd2
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1

MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 8.66 sec   HDFS Read: 11376 HDFS Write: 206 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 5.81 sec   HDFS Read: 6153 HDFS Write: 207 SUCCESS
Total MapReduce CPU Time Spent: 14 seconds 470 msec
OK
US      2023    9       2
UK      2023    9       2
IN      2023    9       2
CA      2023    9       2
AU      2023    9       2
Time taken: 65.93 seconds, Fetched: 5 row(s)
hive>
```

Query 2: Average Session Length Per Week

Logs Inspect Bind mounts Exec Files Stats Debug mode Open in external termin

```
    >     SELECT
    >         user_id,
    >         session_id,
    >         YEAR(CAST(`timestamp` AS TIMESTAMP)) AS year,
    >         WEEKOFYEAR(CAST(`timestamp` AS TIMESTAMP)) AS week,
    >         MAX(UNIX_TIMESTAMP(CAST(`timestamp` AS TIMESTAMP))) AS session_end,
    >         MIN(UNIX_TIMESTAMP(CAST(`timestamp` AS TIMESTAMP))) AS session_start
    >     FROM fact_user_actions
    >     WHERE year = 2023  -- Use partition filtering
    >     GROUP BY user_id, session_id, YEAR(CAST(`timestamp` AS TIMESTAMP)), WEEKOFYEAR(CAST(`timestamp` AS TIMESTAMP))
    > )
    > SELECT
    >     year,
    >     week,
    >     AVG(session_end - session_start) AS avg_session_length
    > FROM session_durations
    > GROUP BY year, week
    > ORDER BY year DESC, week DESC;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execut
ion engine (i.e. spark, tez) or using Hive 1.X releases.
```



5. Pipeline Execution Time

Pipeline Execution Times

Pipeline Stage	Execution Time (seconds)
Data Ingestion (HDFS Upload)	10
Hive Table Creation (raw_metadata)	2
Hive Table Creation (user_activity_logs)	12
Hive Table Creation (fact_user_actions)	0.2
Data Loading into raw_metadata	33
Data Loading into user_activity_logs	23
Data Loading into fact_user_actions	28
Data Transformation into dim_content	23
Data Transformation into fact_user_actions	28

Query Execution Times

Query Description	Execution Time (seconds)
Show Databases	5.8
Show Tables	13.4
Select * from user_activity_logs (LIMIT 10)	15
Select * from raw_metadata (LIMIT 5)	0.24
Insert into fact_user_actions	60.3
Insert into dim_content	23.8
Monthly Active Users by Region	65.9
Top Content by Play Count	82.1

5. Performance Considerations & Optimization

Partitioning

- Used PARTITIONED BY (year, month, day) in fact_user_actions for fast queries.
- Query Optimization: Used partition filters (WHERE year=2023) to avoid scanning the entire table.

File Formats

- Fact Table → Stored as Parquet (efficient columnar storage).
- Dimension Table → Stored as TextFile (simpler, since it's small).

Execution Engine

- Warning: Hive-on-MR is deprecated → Switching to Tez or Spark would improve performance.
- Map-side Join Optimization in Top Categories Query.

6. Conclusion

Designed a Hive ETL pipeline with partitioning, efficient storage formats, and query optimizations. Partition filtering and join optimizations helped improve performance. The total pipeline execution time is around 2-3 minutes, with most queries executing in under 90 seconds.

Github Repository Link:

https://github.com/ShumailaJaved/GROUP_ID_40-Assignment_2_Data_Storage.git