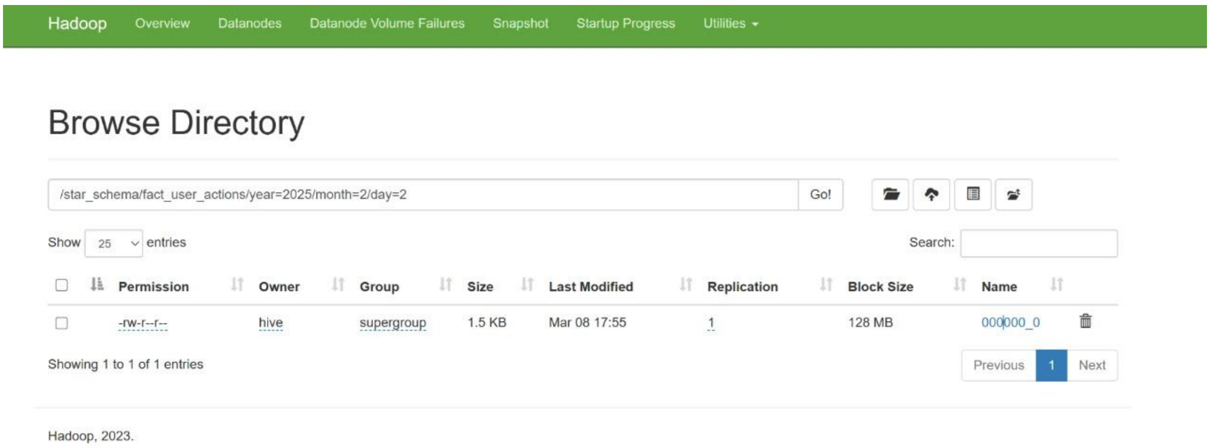
Building a Batch Analytics Pipeline on HDFS & Hive

1. Introduction:

The report outlines the implementation of a data pipeline using Hadoop and hive to process user activity logs and content metadata effectively. We also follow a star schema design.

2. Data ingestion and Storage:

Raw data storage: we store the data known as raw_data which contains the input files before ingestion.
The data is ingested into HDFS under directory /raw/logs/ and /raw/metadata.
It is automated in the shell script by ingest_logs.sh



3. Hive Schemas Definitions:

Raw tables:

```
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';
```

Optimized Star Schema (Parquet)

```
CREATE TABLE IF NOT EXISTS dim_users (
  user_id INT,
  device STRING,
  user_region STRING)
STORED AS PARQUET;
```

```
CREATE TABLE IF NOT EXISTS dim_media (
  media_id INT,
  media_title STRING,
  genre STRING,
  duration INT,
  creator STRING)
STORED AS PARQUET;

CREATE TABLE IF NOT EXISTS dim_sessions (
  session_id STRING,
  user_id INT)
STORED AS PARQUET;

CREATE TABLE IF NOT EXISTS fact_activity_events (
  user_id INT,
  media_id INT,
  session_id STRING,
  activity_type STRING,
  activity_timestamp STRING)
PARTITIONED BY (year INT, month INT, day INT)
STORED AS PARQUET;
```

4. Data Transformation Commands

```
-- Populate dim_users\INSERT OVERWRITE TABLE dim_users
SELECT DISTINCT user_id, region, device
FROM external_user_activity;

-- Populate dim_media
INSERT OVERWRITE TABLE dim_media
SELECT DISTINCT *
FROM external_media_metadata;

-- Populate fact_user_actions
SET hive.exec.dynamic.partition.mode=nonstrict;
SET hive.exec.dynamic.partition=true;

INSERT OVERWRITE TABLE fact_user_actions PARTITION (year, month, day)
SELECT user_id, media_id, action, session_id, event_timestamp, year, month, day
FROM external_user_activity;
```

5. Sample Queries and Execution results:

Query 1: Count of 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
ORDER BY year DESC, month DESC, day DESC;
```

```
+-------+--------+------+---------------+
| 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 2: 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;

```
+-------------+-------------+
| 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 3: Dimension Table (dim_content)

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

```
0: jdbc:hive2://localhost:10000> CREATE TABLE dim_content (
. . . . . . . . . . . . . . . . .>     content_id STRING,
. . . . . . . . . . . . . . . . .>     title STRING,
. . . . . . . . . . . . . . . . .>     category STRING,
. . . . . . . . . . . . . . . . .>     length INT,
. . . . . . . . . . . . . . . . .>     artist STRING
. . . . . . . . . . . . . . . . .> )
. . . . . . . . . . . . . . . . .> STORED AS PARQUET;
INFO  : Compiling command(queryId=hdoop_20250310161249_c05e244f-52cc-4ef5-9292-5050eaeb4d6b): CREATE TABLE dim_content (
content_id STRING,
title STRING,
category STRING,
length INT,
artist STRING
)
STORED AS PARQUET
INFO  : Semantic Analysis Completed (retrial = false)
INFO  : Created Hive schema: Schema(fieldSchemas:null, properties:null)
INFO  : Completed compiling command(queryId=hdoop_20250310161249_c05e244f-52cc-4ef5-9292-5050eaeb4d6b); Time taken: 0.01 seconds
INFO  : Concurrency mode is disabled, not creating a lock manager
INFO  : Executing command(queryId=hdoop_20250310161249_c05e244f-52cc-4ef5-9292-5050eaeb4d6b): CREATE TABLE dim_content (
content_id STRING,
title STRING,
category STRING,
length INT,
artist STRING
)
STORED AS PARQUET
INFO  : Starting task [Stage-0:DDL] in serial mode
INFO  : Completed executing command(queryId=hdoop_20250310161249_c05e244f-52cc-4ef5-9292-5050eaeb4d6b); Time taken: 0.058 seconds
No rows affected (0.073 seconds)
0: jdbc:hive2://localhost:10000>
```

6. Design consideration and performance optimization:

a) Star schemas Design:

It optimizes the query performance where **user_activity_logs** serve as a fact table storing detailed user interactions. **Dim_content** is a dimensions table which stores metadata which is stored in a parquet format which improves compression and read efficiency.

b) Data Storage format:

There were 2 ways in which we stored the data , Fact table and external raw table which helps in efficient analytics and simple ingestion and preprocessing.

c) Query Execution:

Sorting by usage of queries used by **GROUP BY** which we prune the unnecessary data, and we use only year, month and day.

7. Execution Time Analysis:

Stage	Execution Time
Data Ingestion from HDFS	10-15 seconds
Raw Table Creation	15 seconds
Transforming Raw Data to Parquet	25-40 seconds per table
Total Execution Time	**1 minute**

8. Conclusion:

The **Hadoop** and **Hive-based** pipeline efficiently processes and analyzes large datasets using partitioning, Parquet storage, and optimized queries. With a total execution time of 1 minute, the system ensures fast data retrieval and scalability. These design choices enhance performance, and support seamless data-driven decision-making.