Group 7:
Student ID: 24280052 24280069
Student Name: Usmar Haider & Syed Muhammad Mujtaba

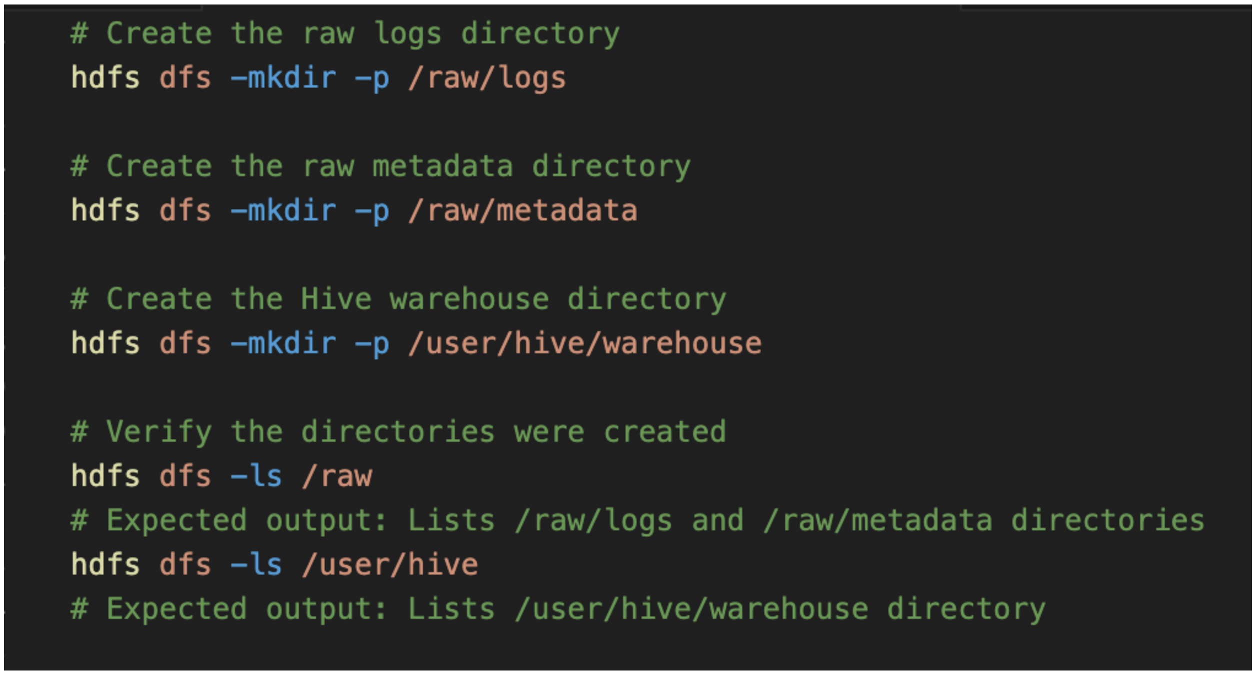
- **This report contains:**

  Scripts:
  - **Step 1: Ingestion Script**
  - **Step 2: Raw Tables in Hive**
  - **Step 3: Star Schema**
  - **Step 4: Transformation**
  - **Step 5: Queries**

Step 1: Ingestion Script:

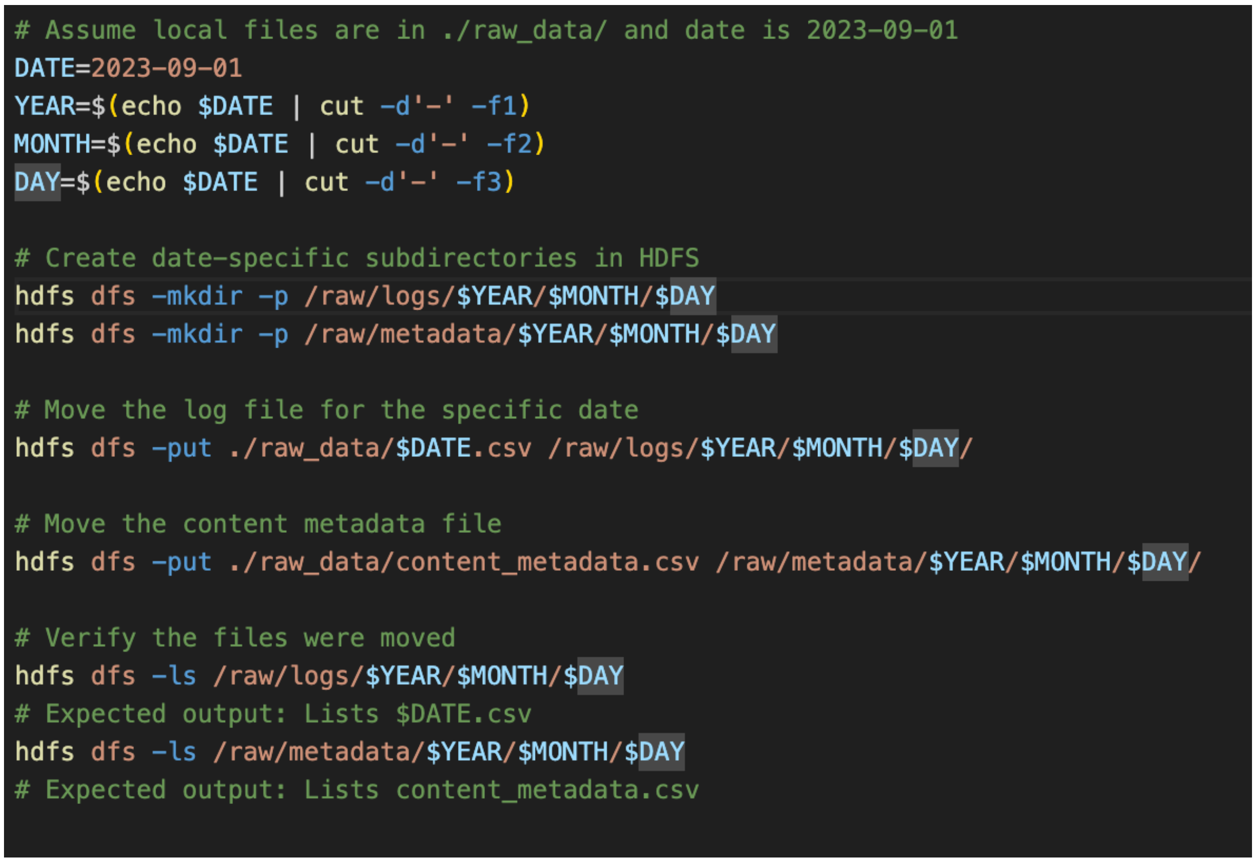
Creating Directories in HDFS

1. This code contains commands to create the necessary directories in HDFS for storing raw logs and metadata, as well as the Hive warehouse.



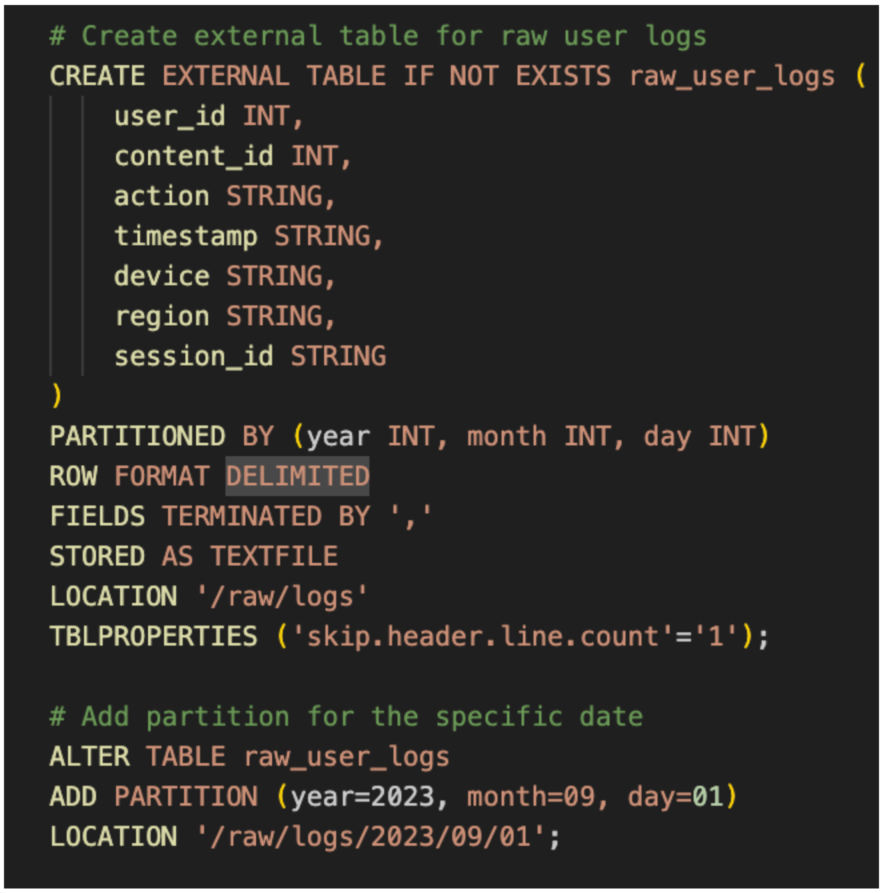
```
# Create the raw logs directory
hdfs dfs -mkdir -p /raw/logs

# Create the raw metadata directory
hdfs dfs -mkdir -p /raw/metadata

# Create the Hive warehouse directory
hdfs dfs -mkdir -p /user/hive/warehouse

# Verify the directories were created
hdfs dfs -ls /raw
# Expected output: Lists /raw/logs and /raw/metadata directories
hdfs dfs -ls /user/hive
# Expected output: Lists /user/hive/warehouse directory
```

2. Moving Files into HDFS



```
# Assume local files are in ./raw_data/ and date is 2023-09-01
DATE=2023-09-01
YEAR=$(echo $DATE | cut -d'-' -f1)
MONTH=$(echo $DATE | cut -d'-' -f2)
DAY=$(echo $DATE | cut -d'-' -f3)

# Create date-specific subdirectories in HDFS
hdfs dfs -mkdir -p /raw/logs/$YEAR/$MONTH/$DAY
hdfs dfs -mkdir -p /raw/metadata/$YEAR/$MONTH/$DAY

# Move the log file for the specific date
hdfs dfs -put ./raw_data/$DATE.csv /raw/logs/$YEAR/$MONTH/$DAY/

# Move the content metadata file
hdfs dfs -put ./raw_data/content_metadata.csv /raw/metadata/$YEAR/$MONTH/$DAY/

# Verify the files were moved
hdfs dfs -ls /raw/logs/$YEAR/$MONTH/$DAY
# Expected output: Lists $DATE.csv
hdfs dfs -ls /raw/metadata/$YEAR/$MONTH/$DAY
# Expected output: Lists content_metadata.csv
```

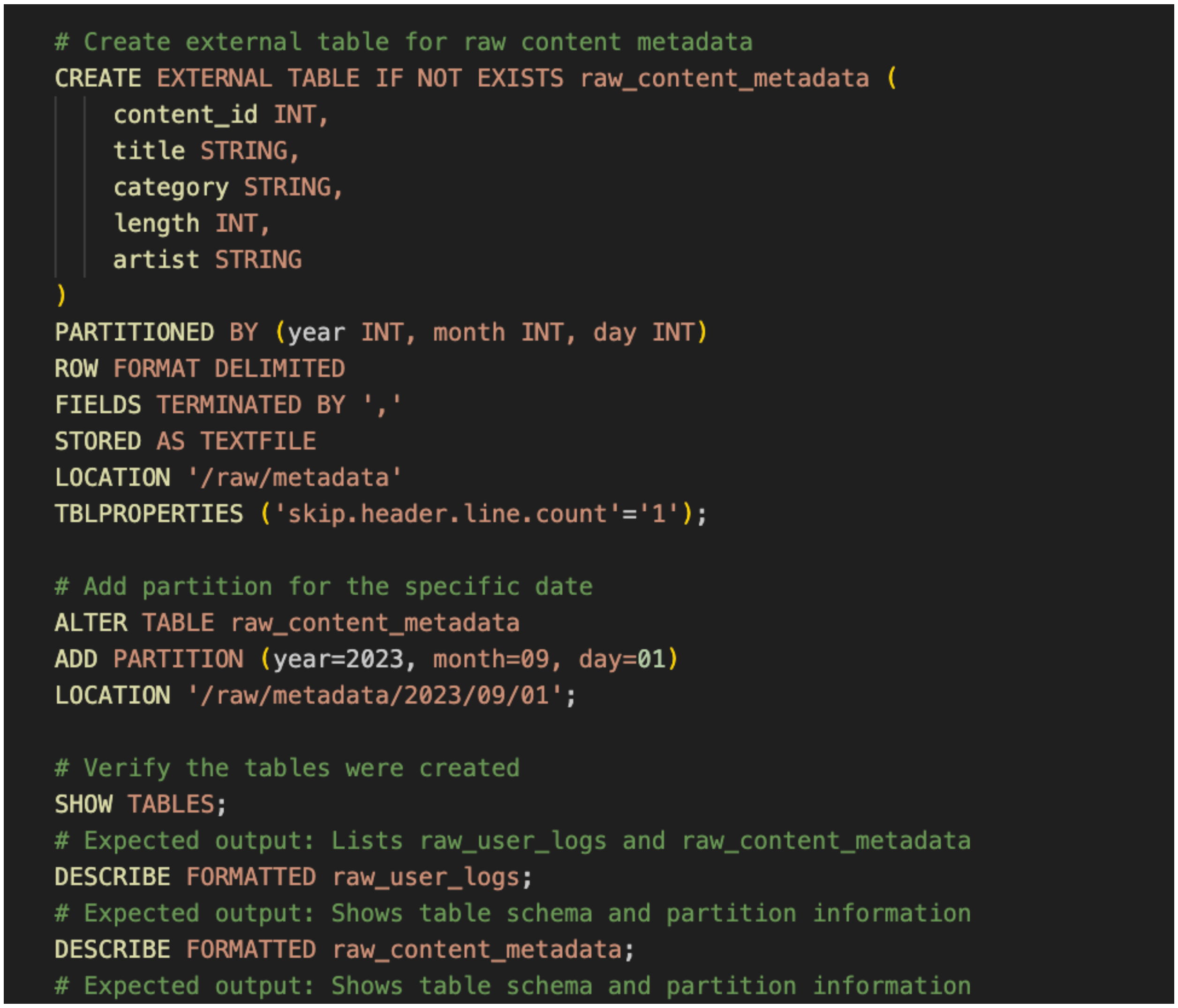
3. Creating Tables in Hive:



```
# Create external table for raw user logs
CREATE EXTERNAL TABLE IF NOT EXISTS 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 DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE
LOCATION '/raw/logs'
TBLPROPERTIES ('skip.header.line.count'='1');

# Add partition for the specific date
ALTER TABLE raw_user_logs
ADD PARTITION (year=2023, month=09, day=01)
LOCATION '/raw/logs/2023/09/01';
```

Step 2:

External Table for raw_content_metadata (Partitioned)

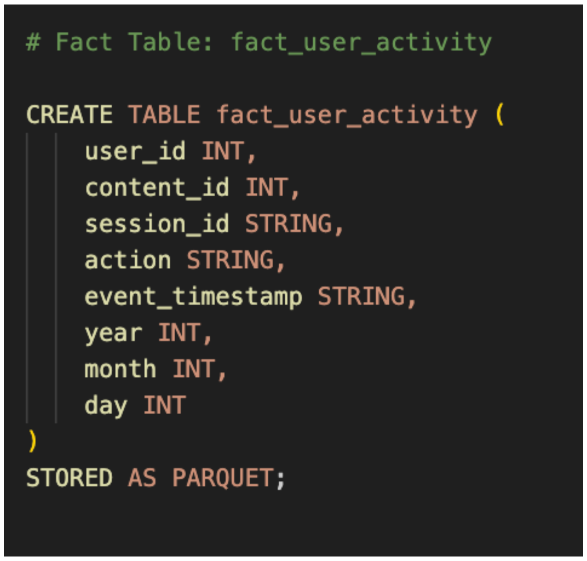


```
# Create external table for raw content metadata
CREATE EXTERNAL TABLE IF NOT EXISTS raw_content_metadata (
    content_id INT,
    title STRING,
    category STRING,
    length INT,
    artist STRING
)
PARTITIONED BY (year INT, month INT, day INT)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE
LOCATION '/raw/metadata'
TBLPROPERTIES ('skip.header.line.count'='1');

# Add partition for the specific date
ALTER TABLE raw_content_metadata
ADD PARTITION (year=2023, month=09, day=01)
LOCATION '/raw/metadata/2023/09/01';

# Verify the tables were created
SHOW TABLES;
# Expected output: Lists raw_user_logs and raw_content_metadata
DESCRIBE FORMATTED raw_user_logs;
# Expected output: Shows table schema and partition information
DESCRIBE FORMATTED raw_content_metadata;
# Expected output: Shows table schema and partition information
```

Step 3: SQL for Tables and Star Schema
Star schema as follows:

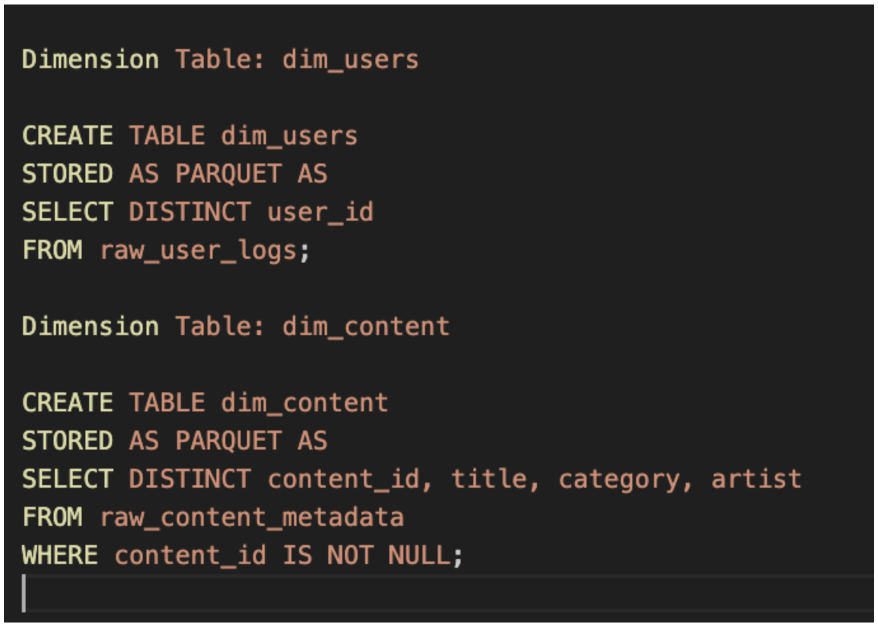
- Fact Table: fact_user_activity
- Dimension Tables:
    - dim_users (user-related details)
    - dim_content (content metadata)
    - dim_sessions (session-related details)

Fact Table: fact_user_activity



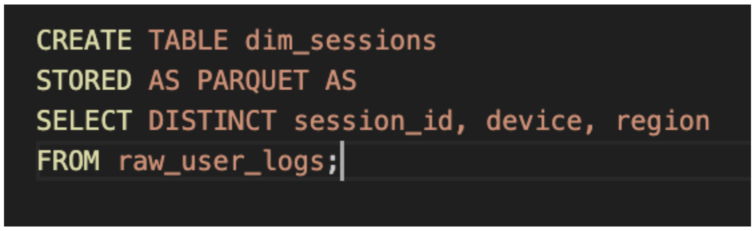
```
# Fact Table: fact_user_activity

CREATE TABLE fact_user_activity (
    user_id INT,
    content_id INT,
    session_id STRING,
    action STRING,
    event_timestamp STRING,
    year INT,
    month INT,
    day INT
)
STORED AS PARQUET;
```

Dimension Tables:



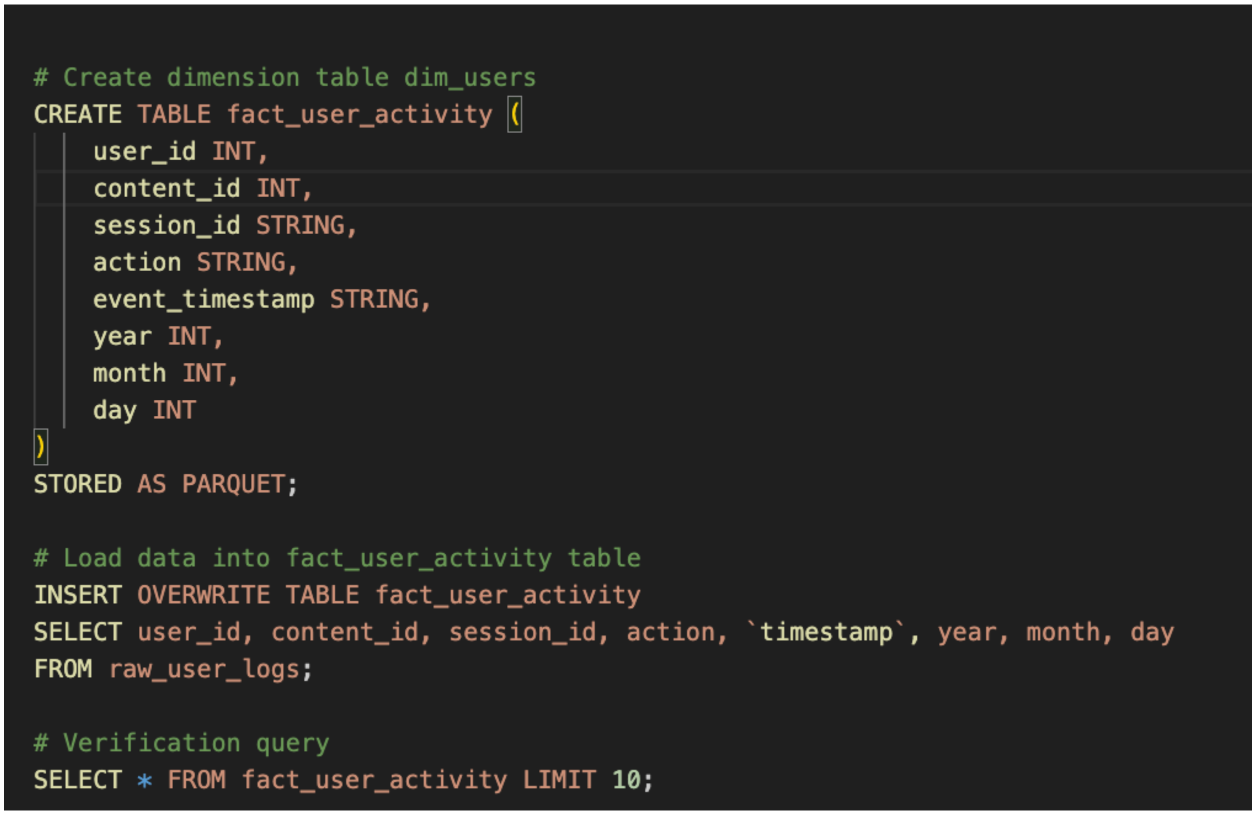
```
Dimension Table: dim_users

CREATE TABLE dim_users
STORED AS PARQUET AS
SELECT DISTINCT user_id
FROM raw_user_logs;

Dimension Table: dim_content

CREATE TABLE dim_content
STORED AS PARQUET AS
SELECT DISTINCT content_id, title, category, artist
FROM raw_content_metadata
WHERE content_id IS NOT NULL;
```

Session Table:



```
CREATE TABLE dim_sessions
STORED AS PARQUET AS
SELECT DISTINCT session_id, device, region
FROM raw_user_logs;
```

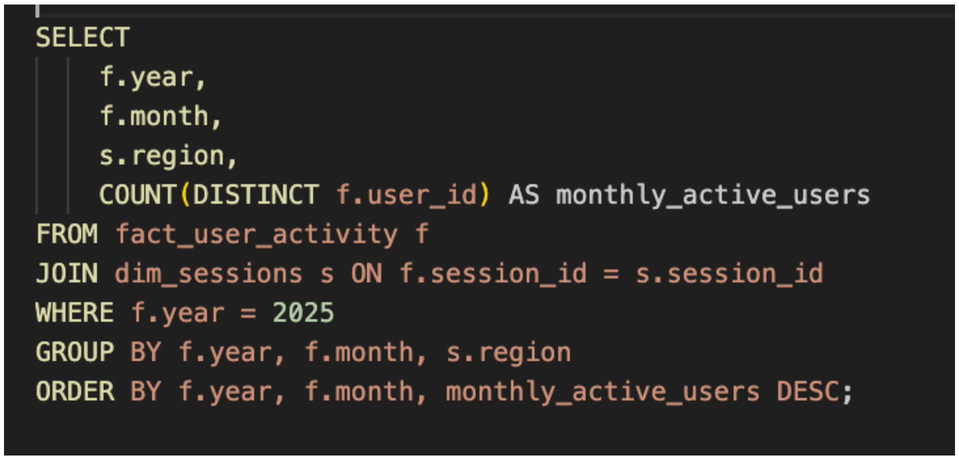
Step 4: Fact Table & Load Data Using INSERT OVERWRITE



```
# Create dimension table dim_users
CREATE TABLE fact_user_activity (
    user_id INT,
    content_id INT,
    session_id STRING,
    action STRING,
    event_timestamp STRING,
    year INT,
    month INT,
    day INT
)
STORED AS PARQUET;

# Load data into fact_user_activity table
INSERT OVERWRITE TABLE fact_user_activity
SELECT user_id, content_id, session_id, action, `timestamp`, year, month, day
FROM raw_user_logs;

# Verification query
SELECT * FROM fact_user_activity LIMIT 10;
```

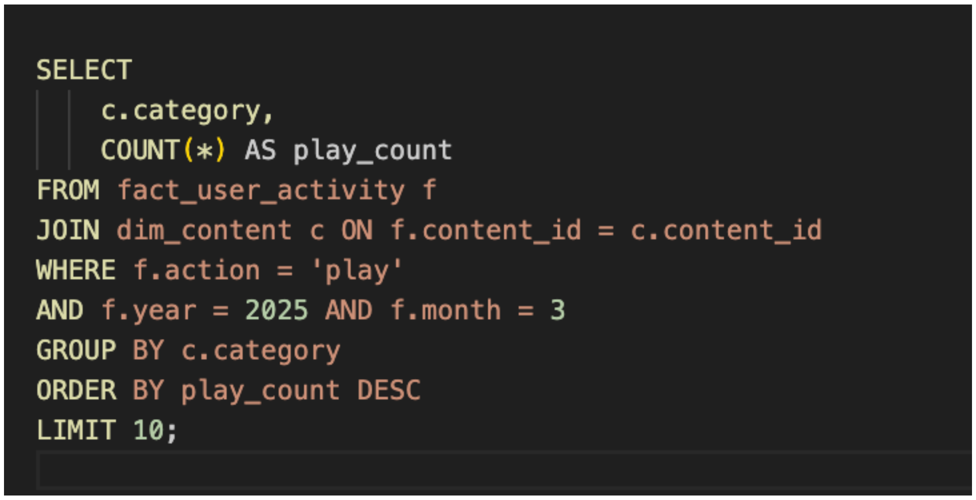
Step 5: Queries

Query 1: Counts Distinct Users Per Region, Per Month



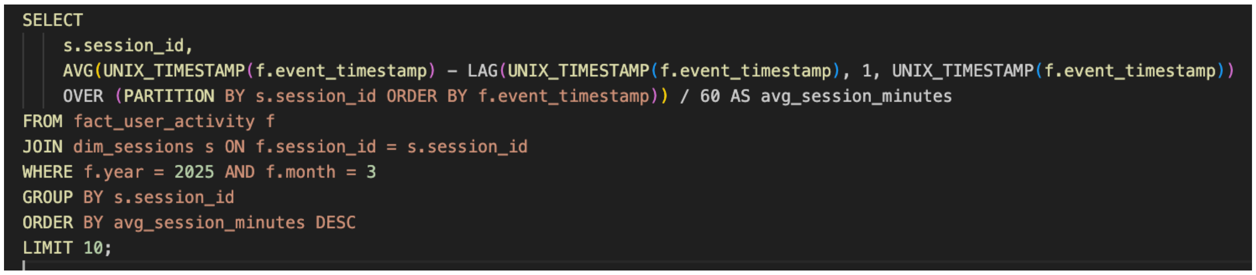
```sql
SELECT
    f.year,
    f.month,
    s.region,
    COUNT(DISTINCT f.user_id) AS monthly_active_users
FROM fact_user_activity f
JOIN dim_sessions s ON f.session_id = s.session_id
WHERE f.year = 2025
GROUP BY f.year, f.month, s.region
ORDER BY f.year, f.month, monthly_active_users DESC;
```

Query 2: Finds the Most-Played Content Categories



```sql
SELECT
    c.category,
    COUNT(*) AS play_count
FROM fact_user_activity f
JOIN dim_content c ON f.content_id = c.content_id
WHERE f.action = 'play'
AND f.year = 2025 AND f.month = 3
GROUP BY c.category
ORDER BY play_count DESC
LIMIT 10;
```

Query 3: Average Session Length Weekly



```sql
SELECT
    s.session_id,
    AVG(UNIX_TIMESTAMP(f.event_timestamp) - LAG(UNIX_TIMESTAMP(f.event_timestamp), 1, UNIX_TIMESTAMP(f.event_timestamp))
    OVER (PARTITION BY s.session_id ORDER BY f.event_timestamp)) / 60 AS avg_session_minutes
FROM fact_user_activity f
JOIN dim_sessions s ON f.session_id = s.session_id
WHERE f.year = 2025 AND f.month = 3
GROUP BY s.session_id
ORDER BY avg_session_minutes DESC
LIMIT 10;
```