Create Hive managed tables

Perquisites before creating Hive Tables

- EMR Cluster Configuration

  The EMR cluster is created with Hive included in the application bundle.

- Streaming Data Processing from Kafka to HDFS via Spark

  Streaming data from Kafka is loaded into the Hadoop Distributed File System (HDFS) using a Spark job.

- Data Transformation for Hive Clickstream Table

  The streaming data loaded into HDFS via Spark is then transformed using another Spark job. This process cleans the data and prepares it for insertion into the Hive clickstream table.

- Batch Data Processing from AWS RDS to HDFS via Scoop

  Batch data from AWS Relational Database Service (RDS) is loaded into HDFS using Scoop, an open-source tool for transferring bulk data between Hadoop and relational databases.

- Loading Batch Data into Hive Bookings Table

  The batch data loaded into HDFS is then transferred to a Hive table named 'bookings’.

- Data Transformation for Date wise Aggregate

  The batch data in HDFS undergoes a transformation via a Spark job. This job performs a date-wise aggregation of the data

- Writing Aggregated Data as CSV for Hive Table

  The aggregated data is written out as a CSV file, which is then used to load data into the Hive table for datewise total bookings.

Steps to Create Hive table

- Open cloud shell and ssh into EMR with key pair with below command

```
ssh -i dev.pem ec2-user@ec2-54-162-87-251.compute-1.amazonaws.com
```

- Run as Hadoop admin with the below commands so we have write permissions in the hdfs

```
sudo su - hdfs
```

- Connect to Hive

  Once you have the Hadoop admin shell open, connect to Hive by simply typing

```
hive
```

```
Create the Hive Table: Execute a CREATE TABLE statement in the next slides to define
the structure of the table. This table structure should match the structure of the
data.
Load Data into the Table: Use a LOAD DATA statement to import the data from HDFS into
your Hive table.
```

Create Hive table for Clickstream Data

```
CREATE TABLE IF NOT EXISTS clickstream (
    customer_id int,
    app_version string,
    os_version string,
    lat double,
    lon double,
    page_id string,
    button_id string,
    is_button_click boolean,
    is_page_view boolean,
    is_scroll_up boolean,
    is_scroll_down boolean,
    `timestamp` timestamp
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
STORED AS TEXTFILE;

Load data inpath '/user/ec2-user/Data/CleanedClickStream/part-00000-84fe7326-3c8d-44aa-
aedc-4417b65f355b-c000.csv' into table clickstream;
```

Create Hive Table for Bookings Data

```
CREATE TABLE IF NOT EXISTS bookings(
            booking_id String,
            customer_id String,
            driver_id String,
            customer_app_version String,
            customer_phone_os_version String,
            pickup_lat String,
            pickup_lon String,
            drop_lat String,
            drop_lon String,
            pickup_timestamp timestamp,
            drop_timestamp timestamp,
            trip_fare float,
            tip_amount float,
            currency_code String,
            cab_color String,
            cab_registration_number String,
            customer_rating_by_driver int,
            rating_by_customer int,
            passenger_count int
      )
     ROW FORMAT DELIMITED
     FIELDS TERMINATED BY ','
     LINES TERMINATED BY '\n'
     STORED AS TEXTFILE;


      Load data inpath '/user/ec2-user/Data/Booking_Batch_Data/part-m-00000' into table bookings;
```

Create Hive Table Date Wise Aggregated Data

```
CREATE TABLE IF NOT EXISTS datewise_total_bookings(
    `date` DATE,
    total_bookings INT
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
STORED AS TEXTFILE;


Load data inpath  '/user/ec2-user/Data/Datewise_Total_Bookings/part-00000-55a0a0ac-
c8db-475c-b4f4-4ea2137a8d93-c000.csv'  into table datewise_total_bookings;
```