Hive Handson

Table of Contents

Note:

Please ensure all `.csv` files are placed in the `bin` folder of Hive. Additionally, all commands should be executed within the terminal session initiated in the `bin` directory.

1. Creating and Using Databases

-- Create a new database

```
CREATE DATABASE pesuniversity;
```

-- Display all databases

```
SHOW DATABASES;
```

-- Switch to the pesuniversity database

```
USE pesuniversity;
```

2. Creating Managed and External Tables

Managed Table:

-- Create a managed table named 'student'

```
CREATE TABLE student (ID INT, Name STRING, Age INT, GPA FLOAT) ROW FORMAT
DELIMITED FIELDS TERMINATED BY ',';
```

-- Describe the structure of the 'student' table

```
DESCRIBE student;
```

-- Detailed description of the 'student' table

```
DESCRIBE FORMATTED student;
```

External Table:

-- Create an external table named 'student_location' with a specified HDFS location

```
CREATE EXTERNAL TABLE student_location (ID INT, Name STRING, Age INT, GPA
FLOAT) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LOCATION '/exp';
```

3. Altering Tables

-- Rename the 'student' table to 'student_table'

```
ALTER TABLE student RENAME TO student_table;
```

-- Add a new column 'Surname' to the 'student_table'

```
ALTER TABLE student_table ADD COLUMNS (Surname STRING);
```

-- Change the column name 'Name' to 'First_Name'

```
ALTER TABLE student_table CHANGE Name First_Name STRING;
```

-- Replace all columns in 'student_table' with a new structure

```
ALTER TABLE student_table REPLACE COLUMNS (ID INT, First_Name STRING, Age
INT, GPA FLOAT, Surname STRING);
```

-- Describe the final structure of 'student_table'

```
DESCRIBE student_table;
```

4. Partitioning

-- Create a new database 'pesstudent'

```
CREATE DATABASE pesstudent;
```

-- Switch to the pesstudent database

```
USE pesstudent;
```

-- Create a partitioned table 'Student' based on the 'Course' column

```
CREATE TABLE Student (ID INT, Name STRING, Age INT)
PARTITIONED BY (Course STRING)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';
```

-- Describe the 'Student' table structure

```
DESCRIBE Student;
```

-- Load data into partitions

```
LOAD DATA LOCAL INPATH 'student.csv' INTO TABLE Student PARTITION
(Course="Big Data");
```

```
LOAD DATA LOCAL INPATH 'student.csv' INTO TABLE Student PARTITION
(Course="DA");
```

```
LOAD DATA LOCAL INPATH 'student.csv' INTO TABLE Student PARTITION
(Course="Graph Theory");
```

-- Query the data where the Course is "DA"

```
SELECT * FROM Student WHERE Course="DA";
```

5. Dynamic Partitioning

-- Create a new database 'pesstudent2'

```
CREATE DATABASE pesstudent2;
```

-- Switch to the pesstudent2 database

```
USE pesstudent2;
```

-- Enable dynamic partitioning

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

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

-- Create a table 'edustudent' to load data

```
CREATE TABLE edustudent (ID INT, Name STRING, Course STRING, Age INT)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';
```

-- Load data into 'edustudent' table

```
LOAD DATA LOCAL INPATH 'student.csv' INTO TABLE edustudent;
```

-- Create a partitioned table 'student_part' and insert data from 'edustudent'

```
CREATE TABLE student_part (ID INT, Name STRING, Age INT)
PARTITIONED BY (Course STRING)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';
```

-- Insert data into partition 'DA'

```
INSERT INTO student_part PARTITION (Course='DA')
SELECT ID, Name, Age, Course FROM edustudent;
```

-- Query all data and data where Course is "DA"

```
SELECT * FROM student_part;
```

```
SELECT * FROM student_part WHERE Course="DA";
```

6. Bucketing

-- Create a new database 'pesbucket'

```
CREATE DATABASE pesbucket;
```

-- Switch to the pesbucket database

```
USE pesbucket;
```

-- Create a basic table 'student_bucket'

```
CREATE TABLE student_bucket (ID INT, Name STRING, Age INT, GPA FLOAT)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';
```

-- Load data into 'student_bucket' table

```
LOAD DATA LOCAL INPATH 'student.csv' INTO TABLE student_bucket;
```

-- Enable bucketing

```
SET hive.enforce.bucketing=true;
```

-- Create a bucketted table 'student_bucketted'

```
CREATE TABLE student_bucketted (ID INT, Name STRING, Age INT, GPA
FLOAT)
CLUSTERED BY (ID) INTO 3 BUCKETS
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';
```

-- Insert data into 'student_bucketted' from 'student_bucket'

```
INSERT OVERWRITE TABLE student_bucketted SELECT * FROM
student_bucket;
```

7. Group By Queries

-- Create a new database 'pesgroup'

```
CREATE DATABASE pesgroup;
```

-- Switch to the pesgroup database

```
USE pesgroup;
```

-- Create a table 'studentgroup' for grouping operations

```
CREATE TABLE studentgroup (ID INT, Name STRING, Age INT, GPA FLOAT,
Country STRING)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';
```

-- Load data into 'studentgroup' table

```
LOAD DATA LOCAL INPATH 'student2.csv' INTO TABLE studentgroup;
```

-- Group by 'Country' and sum the GPA

```
SELECT Country, SUM(GPA) FROM studentgroup GROUP BY Country;
```

-- Group by 'Country' and sum the GPA, only showing results where SUM(GPA) >= 3.5

```
SELECT Country, SUM(GPA) FROM studentgroup GROUP BY Country HAVING
SUM(GPA) >= 3.5;
```

8. Order By and Sort By

-- Create a new database 'pesorder'

```
CREATE DATABASE pesorder;
```

-- Switch to the pesorder database

```
USE pesorder;
```

-- Create a table 'studentorder' for ordering operations

```
CREATE TABLE studentorder (ID INT, Name STRING, Age INT, GPA FLOAT,
Country STRING)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';
```

-- Load data into 'studentorder' table

```
LOAD DATA LOCAL INPATH 'student2.csv' INTO TABLE studentorder;
```

-- Order data by GPA in descending order

```
SELECT * FROM studentorder ORDER BY GPA DESC;
```

-- Sort data by GPA in descending order

```
SELECT * FROM studentorder SORT BY GPA DESC;
```

9. Joins

-- Create a new database 'pesjoin'

```
CREATE DATABASE pesjoin;
```

-- Switch to the pesjoin database

```
USE pesjoin;
```

-- Create the 'student' table with a 'DeptId' column for joining

```
CREATE TABLE student (ID INT, Name STRING, Age INT, GPA FLOAT, COURSE
STRING, DeptId INT)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';
```

-- Load data into the 'student' table

```
LOAD DATA LOCAL INPATH 'student.csv' INTO TABLE student;
```

-- Create the 'department' table with 'DeptId' as the key

```
CREATE TABLE department (DeptId INT, DeptName STRING)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';
```

-- Load data into the 'department' table

```
LOAD DATA LOCAL INPATH 'department.csv' INTO TABLE department;
```

-- Perform a normal join between 'student' and 'department'

```
SELECT s.ID, s.Name, s.Age, s.GPA, s.Course, d.DeptName FROM student
s JOIN department d ON s.deptid = d.deptid;
```

-- Perform a Map Join for efficiency

```
SET hive.auto.convert.join=true;
```

```
SELECT /*+ MAPJOIN(d) */ s.ID, s.Name, s.Age, s.GPA, s.Course,
d.DeptName FROM student s JOIN department d ON s.deptid = d.deptid;
```

10. CRUD Operations with ACID Transactions

-- Enable support for ACID transactions in Hive

```
SET hive.support.concurrency=true;
```

```
SET hive.txn.manager=org.apache.hadoop.hive.ql.lockmgr.DbTxnManager;
```

-- Enable compactor for ACID tables

```
SET hive.compactor.initiator.on=true;
```

```
SET hive.compactor.worker.threads=1;
```

-- Create an ACID table 'books' with transactional properties

```
CREATE TABLE books (
book_id INT,
title STRING,
author STRING,
publication_year INT,
price DOUBLE
)
STORED AS ORC
TBLPROPERTIES ('transactional'='true');
```

-- Insert data into the 'books' table

```
INSERT INTO books (book_id, title, author, publication_year, price)
VALUES
(1, 'The Catcher in the Rye', 'J.D. Salinger', 1951, 10.99),
(2, 'To Kill a Mockingbird', 'Harper Lee', 1960, 7.99),
(3, '1984', 'George Orwell', 1949, 8.99),
(4, 'The Great Gatsby', 'F. Scott Fitzgerald', 1925, 9.99),
(5, 'Moby Dick', 'Herman Melville', 1851, 11.99);
```

-- Update the price of the book with book_id = 1

```
UPDATE books
SET price = 12.99
WHERE book_id = 1;
```

-- Delete the book with book_id = 1

```
DELETE FROM books
WHERE book_id = 1;
```

11. References

[1] Apache Hive
[2] Hive Documentation