1. Create and manage a table with bucketing concept and execute queries with UDFs.

To create and manage a table with the bucketing concept in Hive and execute queries with User Defined Functions (UDFs), follow these steps:

**Step 1: Prepare the Data**

Assume we have a CSV file data.csv with the following content:

1,John,2000

2,Jane,3000

3,Bob,1500

4,Alice,2500

5,Tom,1200

The dataset represents records with the following fields: id, name, and salary.

### Step 2: Create and Load a Bucketed Table

1. **Start Hive**:

hive

**Create a Bucketed Table**:

CREATE TABLE employee\_bucketed (

id INT,

name STRING,

salary DOUBLE

)

CLUSTERED BY (id) INTO 4 BUCKETS

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

STORED AS TEXTFILE;

**Load Data into the Bucketed Table**:

-- Set the following properties to enable dynamic bucketing

SET hive.enforce.bucketing=true;

-- Load data from local storage

LOAD DATA LOCAL INPATH '/path/to/data.csv' INTO TABLE employee\_bucketed;

**Step 3: Verify the Bucketing**

1. **Verify Data Distribution Across Buckets**

-- Query to count the number of records in each bucket

SELECT \*, INPUT\_\_FILE\_\_NAME FROM employee\_bucketed;

### Step 4: Create and Use UDFs

1. **Write a Simple UDF**: Let's create a UDF in Java to calculate a bonus based on salary. For example, we'll give a 10% bonus to each salary.

Create a Java file BonusCalculator.java:

import java.io.IOException;

import org.apache.pig.EvalFunc;

import org.apache.pig.data.Tuple;

public class BonusCalculator extends EvalFunc<Double> {

@Override

public Double exec(Tuple input) throws IOException {

if (input == null || input.size() == 0) {

return null;

}

try {

Double salary = (Double) input.get(0);

return salary \* 0.10;

} catch (Exception e) {

throw new IOException("Caught exception processing input row ", e);

}

}

}

Compile the Java code and package it into a JAR file:

javac -cp /path/to/hive/lib/\*:. BonusCalculator.java

jar -cvf BonusCalculator.jar BonusCalculator.class

**Register the UDF in Hive**:

ADD JAR /path/to/BonusCalculator.jar;

CREATE TEMPORARY FUNCTION bonus\_calculator AS 'BonusCalculator';

**Use the UDF in Queries**:

-- Select all records with the calculated bonus

SELECT id, name, salary, bonus\_calculator(salary) AS bonus FROM employee\_bucketed;

### Complete Commands

Here's the complete sequence of commands to run in the Hive terminal and the shell:

**Hive Terminal:**

-- Create the bucketed table

CREATE TABLE employee\_bucketed (

id INT,

name STRING,

salary DOUBLE

)

CLUSTERED BY (id) INTO 4 BUCKETS

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

STORED AS TEXTFILE;

-- Enable dynamic bucketing

SET hive.enforce.bucketing=true;

-- Load data into the bucketed table

LOAD DATA LOCAL INPATH '/path/to/data.csv' INTO TABLE employee\_bucketed;

-- Verify data distribution across buckets

SELECT \*, INPUT\_\_FILE\_\_NAME FROM employee\_bucketed;

-- Register the UDF

ADD JAR /path/to/BonusCalculator.jar;

CREATE TEMPORARY FUNCTION bonus\_calculator AS 'BonusCalculator';

-- Use the UDF in queries

SELECT id, name, salary, bonus\_calculator(salary) AS bonus FROM employee\_bucketed;

Shell:

# Compile the Java UDF

javac -cp /path/to/hive/lib/\*:. BonusCalculator.java

# Create a JAR file for the UDF

jar -cvf BonusCalculator.jar BonusCalculator.class

Replace /path/to/data.csv with the actual path to your CSV file and /path/to/BonusCalculator.jar with the actual path to your JAR file. This sequence will create a bucketed table, load data into it, register and use a UDF, and execute queries with the UDF.