Create Table Statement

Create Table is a statement used to create a table in Hive. The syntax and example are as follows:

Syntax

CREATE [TEMPORARY] [EXTERNAL] TABLE [IF NOT EXISTS] [db\_name.] table\_name

[(col\_name data\_type [COMMENT col\_comment], ...)]

[COMMENT table\_comment]

[ROW FORMAT row\_format]

[STORED AS file\_format]

Example

Let us assume you need to create a table named **employee** using **CREATE TABLE** statement. The following table lists the fields and their data types in employee table:

|  |  |  |
| --- | --- | --- |
| **Sr.No** | **Field Name** | **Data Type** |
| 1 | Eid | int |
| 2 | Name | String |
| 3 | Salary | Float |
| 4 | Designation | string |

The following data is a Comment, Row formatted fields such as Field terminator, Lines terminator, and Stored File type.

COMMENT ‘Employee details’

FIELDS TERMINATED BY ‘\t’

LINES TERMINATED BY ‘\n’

STORED IN TEXT FILE

The following query creates a table named **employee** using the above data.

hive> CREATE TABLE IF NOT EXISTS employee ( eid int, name String,

salary String, designation String)

COMMENT ‘Employee details’

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ‘\t’

LINES TERMINATED BY ‘\n’

STORED AS TEXTFILE LOCATION ‘/user/rithuik/Emp’;

If you add the option IF NOT EXISTS, Hive ignores the statement in case the table already exists.

On successful creation of table, you get to see the following response:

OK

Time taken: 5.905 seconds

hive>

JDBC Program

The JDBC program to create a table is given example.

import java.sql.SQLException;

import java.sql.Connection;

import java.sql.ResultSet;

import java.sql.Statement;

import java.sql.DriverManager;

public class HiveCreateTable {

private static String driverName = "org.apache.hadoop.hive.jdbc.HiveDriver";

public static void main(String[] args) throws SQLException {

// Register driver and create driver instance

Class.forName(driverName);

// get connection

Connection con = DriverManager.getConnection("jdbc:hive://localhost:10000/userdb", "", "");

// create statement

Statement stmt = con.createStatement();

// execute statement

stmt.executeQuery("CREATE TABLE IF NOT EXISTS "

+" employee ( eid int, name String, "

+" salary String, destignation String)"

+" COMMENT ‘Employee details’"

+" ROW FORMAT DELIMITED"

+" FIELDS TERMINATED BY ‘\t’"

+" LINES TERMINATED BY ‘\n’"

+" STORED AS TEXTFILE;");

System.out.println(“ Table employee created.”);

con.close();

}

}

Save the program in a file named HiveCreateDb.java. The following commands are used to compile and execute this program.

$ javac HiveCreateDb.java

$ java HiveCreateDb

Output

Table employee created.

Load Data Statement

Generally, after creating a table in SQL, we can insert data using the Insert statement. But in Hive, we can insert data using the LOAD DATA statement.

While inserting data into Hive, it is better to use LOAD DATA to store bulk records. There are two ways to load data: one is from local file system and second is from Hadoop file system.

Syntax

The syntax for load data is as follows:

LOAD DATA [LOCAL] INPATH 'filepath' [OVERWRITE] INTO TABLE tablename

[PARTITION (partcol1=val1, partcol2=val2 ...)]

* LOCAL is identifier to specify the local path. It is optional.
* OVERWRITE is optional to overwrite the data in the table.
* PARTITION is optional.

Example

We will insert the following data into the table. It is a text file named**sample.txt** in **/home/user** directory.

1201 Gopal 45000 Technical manager

1202 Manisha 45000 Proof reader

1203 Masthanvali 40000 Technical writer

1204 Kiran 40000 Hr Admin

1205 Kranthi 30000 Op Admin

The following query loads the given text into the table.

hive> LOAD DATA LOCAL INPATH '/home/user/sample.txt'

OVERWRITE INTO TABLE employee;

On successful download, you get to see the following response:

OK

Time taken: 15.905 seconds

hive>

JDBC Program

Given below is the JDBC program to load given data into the table.

import java.sql.SQLException;

import java.sql.Connection;

import java.sql.ResultSet;

import java.sql.Statement;

import java.sql.DriverManager;

public class HiveLoadData {

private static String driverName = "org.apache.hadoop.hive.jdbc.HiveDriver";

public static void main(String[] args) throws SQLException {

// Register driver and create driver instance

Class.forName(driverName);

// get connection

Connection con = DriverManager.getConnection("jdbc:hive://localhost:10000/userdb", "", "");

// create statement

Statement stmt = con.createStatement();

// execute statement

stmt.executeQuery("LOAD DATA LOCAL INPATH '/home/user/sample.txt'" + "OVERWRITE INTO TABLE employee;");

System.out.println("Load Data into employee successful");

con.close();

}

}

Save the program in a file named HiveLoadData.java. Use the following commands to compile and execute this program.

$ javac HiveLoadData.java

$ java HiveLoadData

Output:

Load Data into employee successful

## Alter Table Statement

It is used to alter a table in Hive.

### Syntax

The statement takes any of the following syntaxes based on what attributes we wish to modify in a table.

ALTER TABLE name RENAME TO new\_name

ALTER TABLE name ADD COLUMNS (col\_spec[, col\_spec ...])

ALTER TABLE name DROP [COLUMN] column\_name

ALTER TABLE name CHANGE column\_name new\_name new\_type

ALTER TABLE name REPLACE COLUMNS (col\_spec[, col\_spec ...])

## Rename To… Statement

The following query renames the table from **employee** to **emp**.

hive> ALTER TABLE employee RENAME TO emp;

### JDBC Program

The JDBC program to rename a table is as follows.

import java.sql.SQLException;

import java.sql.Connection;

import java.sql.ResultSet;

import java.sql.Statement;

import java.sql.DriverManager;

public class HiveAlterRenameTo {

private static String driverName = "org.apache.hadoop.hive.jdbc.HiveDriver";

public static void main(String[] args) throws SQLException {

// Register driver and create driver instance

Class.forName(driverName);

// get connection

Connection con = DriverManager.getConnection("jdbc:hive://localhost:10000/userdb", "", "");

// create statement

Statement stmt = con.createStatement();

// execute statement

stmt.executeQuery("ALTER TABLE employee RENAME TO emp;");

System.out.println("Table Renamed Successfully");

con.close();

}

}

Save the program in a file named HiveAlterRenameTo.java. Use the following commands to compile and execute this program.

$ javac HiveAlterRenameTo.java

$ java HiveAlterRenameTo

### Output:

Table renamed successfully.

## Change Statement

The following table contains the fields of **employee** table and it shows the fields to be changed (in bold).

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Convert from Data Type** | **Change Field Name** | **Convert to Data Type** |
| eid | Int | eid | int |
| **name** | String | **ename** | String |
| salary | **Float** | salary | **Double** |
| designation | String | designation | String |

The following queries rename the column name and column data type using the above data:

hive> ALTER TABLE employee CHANGE name ename String;

hive> ALTER TABLE employee CHANGE salary salary Double;

### JDBC Program

Given below is the JDBC program to change a column.

import java.sql.SQLException;

import java.sql.Connection;

import java.sql.ResultSet;

import java.sql.Statement;

import java.sql.DriverManager;

public class HiveAlterChangeColumn {

private static String driverName = "org.apache.hadoop.hive.jdbc.HiveDriver";

public static void main(String[] args) throws SQLException {

// Register driver and create driver instance

Class.forName(driverName);

// get connection

Connection con = DriverManager.getConnection("jdbc:hive://localhost:10000/userdb", "", "");

// create statement

Statement stmt = con.createStatement();

// execute statement

stmt.executeQuery("ALTER TABLE employee CHANGE name ename String;");

stmt.executeQuery("ALTER TABLE employee CHANGE salary salary Double;");

System.out.println("Change column successful.");

con.close();

}

}

Save the program in a file named HiveAlterChangeColumn.java. Use the following commands to compile and execute this program.

$ javac HiveAlterChangeColumn.java

$ java HiveAlterChangeColumn

### Output:

Change column successful.

## Add Columns Statement

The following query adds a column named dept to the employee table.

hive> ALTER TABLE employee ADD COLUMNS (

dept STRING COMMENT 'Department name');

## JDBC Program

The JDBC program to add a column to a table is given below.

import java.sql.SQLException;

import java.sql.Connection;

import java.sql.ResultSet;

import java.sql.Statement;

import java.sql.DriverManager;

public class HiveAlterAddColumn {

private static String driverName = "org.apache.hadoop.hive.jdbc.HiveDriver";

public static void main(String[] args) throws SQLException {

// Register driver and create driver instance

Class.forName(driverName);

// get connection

Connection con = DriverManager.getConnection("jdbc:hive://localhost:10000/userdb", "", "");

// create statement

Statement stmt = con.createStatement();

// execute statement

stmt.executeQuery("ALTER TABLE employee ADD COLUMNS " + " (dept STRING COMMENT 'Department name');");

System.out.prinln("Add column successful.");

con.close();

}

}

Save the program in a file named HiveAlterAddColumn.java. Use the following commands to compile and execute this program.

$ javac HiveAlterAddColumn.java

$ java HiveAlterAddColumn

### Output:

Add column successful.

## Replace Statement

The following query deletes all the columns from the **employee** table and replaces it with **emp** and **name** columns:

hive> ALTER TABLE employee REPLACE COLUMNS (

eid INT empid Int,

ename STRING name String);

## JDBC Program

Given below is the JDBC program to replace **eid** column with **empid** and**ename**column with **name**.

import java.sql.SQLException;

import java.sql.Connection;

import java.sql.ResultSet;

import java.sql.Statement;

import java.sql.DriverManager;

public class HiveAlterReplaceColumn {

private static String driverName = "org.apache.hadoop.hive.jdbc.HiveDriver";

public static void main(String[] args) throws SQLException {

// Register driver and create driver instance

Class.forName(driverName);

// get connection

Connection con = DriverManager.getConnection("jdbc:hive://localhost:10000/userdb", "", "");

// create statement

Statement stmt = con.createStatement();

// execute statement

stmt.executeQuery("ALTER TABLE employee REPLACE COLUMNS "

+" (eid INT empid Int,"

+" ename STRING name String);");

System.out.println(" Replace column successful");

con.close();

}

}

Save the program in a file named HiveAlterReplaceColumn.java. Use the following commands to compile and execute this program.

$ javac HiveAlterReplaceColumn.java

$ java HiveAlterReplaceColumn

### Output:

Replace column successful.

Create database xxx;

// For create database

Show databases

// For view list of databases

Use xxx;

// For connecting to specific database

Use default;

// For connecting to default database (already existed one for coming back)

Drop database xxx;

// For droping database