Big Data And Hadoop

Session 16 – Assignment 2

Problem Statement:

Write a hive UDF that implements functionality of string concat_ws(string SEP, array<string>).
This UDF will accept two arguments, one string and one array of string.
It will return a single string where all the elements of the array are separated by the SEP.

Solution:

Step 1: The input file is present on the local file system at **/home/acadgild/Abhilasha/hive** as follows:

```
acadgild@localhost:~/Abhilasha/hive
File  Edit  View  Search  Terminal  Help
[acadgild@localhost hive]$ pwd
/home/acadgild/Abhilasha/hive
[acadgild@localhost hive]$ ls -l
total 600
-rw-rw-r--. 1 acadgild acadgild   6339 Oct  8 19:16 commands
-rw-rw-r--. 1 acadgild acadgild   6309 Oct  8 18:28 commands~
-rw-rw-r--. 1 acadgild acadgild    170 Sep 17 14:17 complexData
-rw-rw-r--. 1 acadgild acadgild    437 Sep 16 19:29 dataset_Session14.txt
-rw-rw-r--. 1 acadgild acadgild    159 Sep 19 08:49 emp_Details
-rw-rw-r--. 1 acadgild acadgild    159 Sep 19 08:24 emp_Details~
-rw-rw-r--. 1 acadgild acadgild     84 Sep 17 13:43 empDetails~
-rw-rw-r--. 1 acadgild acadgild    107 Sep 18 22:00 employee.csv
-rw-rw-r--. 1 acadgild acadgild    107 Sep 18 21:51 employee.csv~
-rw-rw-r--. 1 acadgild acadgild    282 Oct  2 18:17 Emp_Sal
-rw-rw-r--. 1 acadgild acadgild      0 Oct  2 18:13 Emp_Sal~
-rw-rw-r--. 1 acadgild acadgild     43 Oct  1 19:36 locations
-rw-rw-r--. 1 acadgild acadgild     43 Oct  1 19:36 locations~
-rw-rw-r--. 1 acadgild acadgild     75 Oct  5 07:10 olympic
-rw-rw-r--. 1 acadgild acadgild      0 Oct  5 07:08 olympic~
-rw-rw-r--. 1 acadgild acadgild 518669 Sep 19 22:14 olympix_data.csv
drwxrwxr-x. 2 acadgild acadgild   4096 Sep 19 08:53 output
drwxrwxr-x. 2 acadgild acadgild   4096 Sep 19 22:59 output-Query3
drwxrwxr-x. 2 acadgild acadgild   4096 Sep 19 22:55 output-Query4
-rw-rw-r--. 1 acadgild acadgild   2547 Oct  8 18:59 udf1.jar
-rw-rw-r--. 1 acadgild acadgild   2540 Oct  8 19:09 udf.jar
-rw-rw-r--. 1 acadgild acadgild    170 Sep 17 14:17 Unsaved Document 1~
-rw-rw-r--. 1 acadgild acadgild     97 Oct  1 19:35 users
-rw-rw-r--. 1 acadgild acadgild     85 Oct  1 19:34 users~
[acadgild@localhost hive]$
```

Step 2: The content of the input file can be seen using the **cat** command as follows:

```
acadgild@localhost:~/Abhilasha/hive
File  Edit  View  Search  Terminal  Help
[acadgild@localhost hive]$ cat olympic;
1       Abigel   Swimming,Badminton
2       Marry    JavelianThrow,Running
3       Joby     Badminton
[acadgild@localhost hive]$
```

Step 3: We start the hive command line by executing the command hive as shown below:

```
acadgild@localhost:~
File  Edit  View  Search  Terminal  Help
[acadgild@localhost ~]$ hive

Logging initialized using configuration in jar:file:/usr/local/hive/lib/hive-common-0.14.0.jar!/hive-log4j.properties
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hive/lib/hive-jdbc-0.14.0-standalone.jar!/org/slf4j/impl/StaticLoggerBinder.clas
s]
SLF4J: Found binding in [jar:file:/usr/local/hadoop-2.6.0/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/Sta
ticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
hive>
```

The above snapshot also shows that hive prompt has started. A pre-requisite to use hive is to start mysql server. This was done using the command sudo service mysqld start.

Step 4: We use **SHOW DATABASES** command to list the databases present. The database we will be using is **custom** as shown below:

```
acadgild@localhost:~
File  Edit  View  Search  Terminal  Help

hive> SHOW DATABASES;
OK
acadgild
b1
custom
default
Time taken: 0.067 seconds, Fetched: 4 row(s)
hive>
```

Step 5: We use **USE custom** command to make use of custom database, as shown below:

```
acadgild@localhost:~
File  Edit  View  Search  Terminal  Help

hive> USE custom;
OK
Time taken: 0.036 seconds
hive>
hive> USE custom;
OK
Time taken: 0.036 seconds
hive>
```

Step 6: We create the table using **CREATE TABLE** command. The fields of the table are: id, name, and sportName.

```
acadgild@localhost:~
File  Edit  View  Search  Terminal  Help

hive> CREATE TABLE olympic
(
    id INT,
    name STRING,
    sportName array<String>
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
collection items terminated by ',';
OK
Time taken: 0.187 seconds
hive>
```

Here, we mention that the column data in the file is separated by tab and the third column being list, its values are separated by ‘,’ in the data file.

Step 7: **DESCRIBE** command will help us verify the schema of the table as follows:

```
acadgild@localhost:~
File  Edit  View  Search  Terminal  Help

hive> DESCRIBE olympic;
OK
id                      int
name                    string
sportname               array<string>
Time taken: 0.213 seconds, Fetched: 3 row(s)
hive>
acadgild@localhost:~ | [hive - File Browser] | *commands (~/Abhila... | Java - Hive-UDF/src/St...
```

Id is of type integer. Name is a string and sportname is an array list of strings.

Step 8: Next is to load the data from input file, which is located at **/home/acadgild/Abhilasha/hive** as follows. We use the **LOAD** command and use the keyword **LOCAL** to specify that the file is present in the local file system and not HDFS.

```
acadgild@localhost:~
File  Edit  View  Search  Terminal  Help

hive> LOAD DATA LOCAL INPATH '/home/acadgild/Abhilasha/hive/olympic'
INTO TABLE olympic;
Loading data to table custom.olympic
Table custom.olympic stats: [numFiles=1, totalSize=75]
OK
Time taken: 1.522 seconds
hive>
acadgild@localhost:~ | [hive - File Browser] | commands (~/Abhilas... | Java - Hive-UDF/src/St...
```

Step 9: Using the **SELECT *** query, we can display the complete data as follows:

```
acadgild@localhost:~
File  Edit  View  Search  Terminal  Help

hive> SELECT * FROM olympic;
OK
1       Abigel   ["Swimming","Badminton"]
2       Marry    ["JavelianThrow","Running"]
3       Joby     ["Badminton"]
Time taken: 0.488 seconds, Fetched: 3 row(s)
hive>
```

Step 10: In order to use a UDF, we need to generate a jar of our code and register that jar in hive's context. The code file used is **StringConcat.java**. Its jar file is created with the name **udf.jar** and placed at **/home/acadgild/Abhilasha/hive.**

Step 11: We can add the jar in hive's build path as follows:

```
acadgild@localhost:~
File  Edit  View  Search  Terminal  Help

hive> ADD JAR /home/acadgild/Abhilasha/hive/udf.jar;
Added [/home/acadgild/Abhilasha/hive/udf.jar] to class path
Added resources: [/home/acadgild/Abhilasha/hive/udf.jar]
hive>
```

Step 12: To verify the addition of jar, we can use **list jars** command as follows:

```
acadgild@localhost:~
File  Edit  View  Search  Terminal  Help

hive> list jars;
/home/acadgild/Abhilasha/hive/udf.jar
hive>
```

Step 13: Now, we need to create a temporary function that will be used while executing hive query. This can be done as follows:

```
acadgild@localhost:~
File  Edit  View  Search  Terminal  Help

hive> CREATE TEMPORARY FUNCTION concat_ws AS 'StringConcat';
OK
Time taken: 0.011 seconds
hive>
```

Step 14: Now, we are all set to use the udf. This udf will concatenate all the values in the array, using the separator mentioned as the first parameter to this udf. We are using the separator '-' and will concatenate all the values in the column **sportname** as follows:

```
acadgild@localhost:~
File  Edit  View  Search  Terminal  Help

hive> SELECT id, name, concat_ws("-",sportName) FROM olympic;
OK
1       Abigel  Swimming-Badminton
2       Marry   JavelianThrow-Running
3       Joby    Badminton
Time taken: 0.179 seconds, Fetched: 3 row(s)
hive>
```

The above snapshot also mentions the output of the query containing the execution of udf. The values of column sportname are now separated by '-'.