**UDTF**: User defined tabular function works on one row as input and returns multiple rows as output.

So here the relation is one to many. e.g Hive built in EXPLODE() function.

Now let’s take an array column USER\_IDS as ARRAY10, 12, 5, 45>

Then SELECT EXPLODE (USER\_IDS) as ID FROM T\_USER

Will give 10, 12,5,45 as four different rows in output.

UDTF can be used to split a column into multiple column as well which we will look in below example. Here alias "AS" clause is mandatory**.**

**books.data**

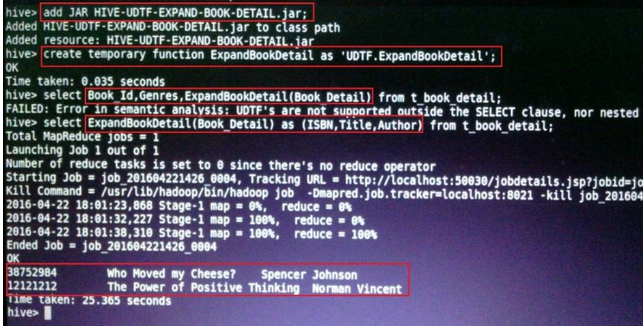
|  |
| --- |
| 1001 6438391,book1-title,Author1  1002 5288292,book2-title,Author2  1003 6878195,book3-title,Author3 |

Table Script

|  |
| --- |
| create table books (book\_id string,  book\_detail string)  row format delimited  fields terminated by "\t"; |

Load data into table

|  |
| --- |
| load data local inpath '/home/mac127/test/student.data' into table student; |



**Problem Statement:** Find the Mean of Marks obtained in Math by all of the students

Create UDAF class

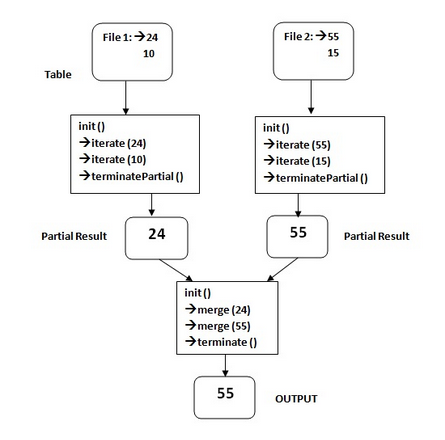
|  |
| --- |
| **package** org.cyb.UDAF;  **public** **class** PartialResult {  **double** result;  **int** count;  } |

|  |
| --- |
| **package** org.cyb.UDAF;  **import** org.apache.hadoop.hive.ql.exec.UDAF;  **import** org.apache.hadoop.hive.ql.exec.UDAFEvaluator;  **import** org.apache.hadoop.hive.serde2.io.DoubleWritable;  **public** **class** GetMeanMarks **extends** UDAF{  **public** **static** **class** GetIntMeanEvaluator **implements** UDAFEvaluator  {    PartialResult part;    **public** **void** init()  {  part = **null**;  }    **public** **boolean** iterate(DoubleWritable value)  {  **if** (value == **null**)  {  **return** **true**;  }  **if**(part == **null**)  {  part = **new** PartialResult();  }  part.result = part.result + value.get();  part.count++;  **return** **true**;  }    **public** PartialResult terminatePartial()  {  **return** part;  }    **public** **boolean** merge(PartialResult otherFile)  {  **if**(otherFile == **null**)  {  **return** **true**;  }  **if**(part == **null**)  {  part= **new** PartialResult();  }  part.result = part.result + otherFile.result;  part.count = part.count + otherFile.count;  **return** **true**;  }    **public** DoubleWritable terminate()  {  **if** ( part == **null**)  {  **return** **null**;  }  **return** **new** DoubleWritable( (part.result)/part.count);  }  }  } |

Create HIVE\_UDAF\_MEANMARKS.jar, please find attached



Below is the sample flow diagram to explain the UDAF methods:



Now add jar and create function in hive

|  |
| --- |
| hive> add jar /home/mac127/test/HIVE\_UDAF\_MEANMARKS.jar;  hive> CREATE TEMPORARY FUNCTION GetMeanMarks AS 'org.cyb.UDAF.GetMeanMarks'; |

Run below query

|  |
| --- |
| select GetMeanMarks(math) as mathmean from student; |

|  |
| --- |
| hive> add jar /home/mac127/test/HIVE\_UDAF\_MEANMARKS.jar;  Added [/home/mac127/test/HIVE\_UDAF\_MEANMARKS.jar] to class path  Added resources: [/home/mac127/test/HIVE\_UDAF\_MEANMARKS.jar] |
| hive> CREATE TEMPORARY FUNCTION GetMeanMarks AS 'org.cyb.UDAF.GetMeanMarks';  OK  Time taken: 0.014 seconds |
| hive> select GetMeanMarks(math) as mathmean from student;  Query ID = root\_20170224180909\_c2feacdf-d634-4bbf-94aa-a0ba900bc470  Total jobs = 1  Launching Job 1 out of 1  Number of reduce tasks determined at compile time: 1  In order to change the average load for a reducer (in bytes):  set hive.exec.reducers.bytes.per.reducer=<number>  In order to limit the maximum number of reducers:  set hive.exec.reducers.max=<number>  In order to set a constant number of reducers:  set mapreduce.job.reduces=<number>  Starting Job = job\_1484320332413\_0144, Tracking URL = http://mac127:8088/proxy/application\_1484320332413\_0144/  Kill Command = /opt/cloudera/parcels/CDH-5.9.0-1.cdh5.9.0.p0.23/lib/hadoop/bin/hadoop job -kill job\_1484320332413\_0144  Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1  2017-02-24 18:09:42,899 Stage-1 map = 0%, reduce = 0%  2017-02-24 18:09:50,334 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 2.16 sec  2017-02-24 18:09:59,720 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 4.86 sec  MapReduce Total cumulative CPU time: 4 seconds 860 msec  Ended Job = job\_1484320332413\_0144  MapReduce Jobs Launched:  Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 4.86 sec HDFS Read: 7486 HDFS Write: 5 SUCCESS  Total MapReduce CPU Time Spent: 4 seconds 860 msec  OK  79.2  Time taken: 29.85 seconds, Fetched: 1 row(s) |
| Check result with existing avg function  hive> select avg(math) as mathmean from student;  Query ID = root\_20170224181010\_33aff01c-ef53-439b-9e69-31b1ffa0a474  Total jobs = 1  Launching Job 1 out of 1  Number of reduce tasks determined at compile time: 1  In order to change the average load for a reducer (in bytes):  set hive.exec.reducers.bytes.per.reducer=<number>  In order to limit the maximum number of reducers:  set hive.exec.reducers.max=<number>  In order to set a constant number of reducers:  set mapreduce.job.reduces=<number>  Starting Job = job\_1484320332413\_0145, Tracking URL = http://mac127:8088/proxy/application\_1484320332413\_0145/  Kill Command = /opt/cloudera/parcels/CDH-5.9.0-1.cdh5.9.0.p0.23/lib/hadoop/bin/hadoop job -kill job\_1484320332413\_0145  Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1  2017-02-24 18:10:57,799 Stage-1 map = 0%, reduce = 0%  2017-02-24 18:11:05,077 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 1.63 sec  2017-02-24 18:11:13,403 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 3.67 sec  MapReduce Total cumulative CPU time: 3 seconds 670 msec  Ended Job = job\_1484320332413\_0145  MapReduce Jobs Launched:  Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 3.67 sec HDFS Read: 7414 HDFS Write: 5 SUCCESS  Total MapReduce CPU Time Spent: 3 seconds 670 msec  OK  79.2  Time taken: 27.333 seconds, Fetched: 1 row(s)  hive> |