UDFs in Pig

|  |  |
| --- | --- |
| **Exercise Dir** | ~/workspace/pig\_udfs |
| **Eclipse Proj** | pig\_udfs |
| **JAR File** | partitioner.jar |

|  |  |
| --- | --- |
| **Objective:** | Write a Pig User Defined Function (UDF). |
| **Location of Files:** | ~/materials/data |
| **Successful Outcome:** | A Pig script the computes the on-balance volume of a specified stock. |
| **Before You Begin:** | Start Eclipse in your lab instance |
| **Eclipse Project** | pig\_udfs |
| **Exercise directory** | ~/workspace/pig\_udfs |

**Define an EvalFunc Class**

1.   In Eclipse, locate the pig\_udfsproject.

2.   Add a new class to the stockudfs package named OnBalanceVolume that extends the EvalFunc class.

3.   Change the generic of OnBalanceVolume to Long.

4.   Add two fields to OnBalanceVolume:

`

 5.   Add the exec method:

@Overridepublic Long exec(Tuple input) throws IOException {}

6.   Save the input parameters into local variables by parsing the first parameter to a long (which is the volume) and the second parameter to a double (which is the stock’s closing price).

7.   Have the exec method return the On-Balance Volume (OBV) of the stock, which is computed from the following formula:

1.      OBV = previous OBV + volume (if current close > previous close)

2.      OBV = previous OBV - volume (if current close < previous close)

3.      OBV = previous OBV + 0 (if current close == previous close)

8. Before you return the OBV from exec, make sure you save the previous OBV and stock prices in the respective fields.

9.Save your changes to OnBalanceVolume.java.

10. Deploy the JAR.

11.Build the project to create the stockudfs.jarfile.

12. Write the Pig Script.

13.Create a new text file in the Pig project in Eclipse named stockvolume.pig.

14. Using the register command, register stockudfs.jar:

register /home/[username]/workspace/pig\_udfs/ stockudfs.jar;

15. Load the stock prices from the NYSE\_daily\_prices\_A.cvsfile in HDFS:

stockdata = LOAD ‘stocks/NYSE\_daily\_prices\_A.csv' using PigStorage(',') AS(exchange:chararray,symbol:chararray, date:chararray,open:float,high:float,low:float,

close:float,volume:int);

16. Create a group that contains the stock’s symbol, date, volume and closing price:

stocks\_all = FOREACH stockdata GENERATE symbol, date, close, volume;

17. Filter the group by the symbol specified from the command line:

stocks\_filter = FILTER stocks\_all BY symbol == '$symbol';

18. Order the remaining stocks by date ascending:

stocks\_sorted = ORDER stocks\_filter BY date ASC;

19. Use the OnBalanceVolume UDF to compute the on-balance volume of the specified stock:

obv\_result = FOREACH stocks\_sorted GENERATE symbol, date, stockudfs.OnBalanceVolume(volume, close) AS obv;

20. Dump the results:

dump obv\_result;

21.Save your changes to stockvolume.pig.

22.Run the Pig script, using “AVA” as the symbol parameter:

$ cd ~/workspace/pig\_udfs/

$ pig -param symbol=AVA stockvolume.pig

23. The results should include the stock symbol, date, and the on-balance volume of the stock on that particular date. For example, the ending output of “AVA” should look like:

(AVA,2010-02-02,52525800)

(AVA,2010-02-03,52336000)  
 (AVA,2010-02-04,52184800)  
 (AVA,2010-02-05,51843300)  
 (AVA,2010-02-08,51636000)

Result: This lab uses Pig to compute the on-balance volume of a collection of stock prices. You used the DUMP command to verify the task, but you could have used a STORE command and saved the results to files in HDFS.

**END**