Weather assignment

Explanation of the Dataset

Here we have taken a one row as the sample dataset to explain the various columns required in processing of the problem statement.

63891 20130101 5.102 -86.61 32.85 12.8 9.6 11.2 11.6 19.4 -9999.00 U -9999.0 -9999.0 -9999.0 -9999.0 -9999.0 -9999.0 -99.000 -99.000 -99.000 -99.000 -99.000 -9999.0 -9999.0 -9999.0 -9999.0 -9999.0

In the first column having the value 63891,the position where 6 is present is termed as index 0 and the 1st column ends at the index value 5 and the second column i.e the the date starts from the index value 6 and ends at index value 14

In the same way the the sixth column i.e maximum temperature column starts from index value 38 and ends at index value 45

The seventh column i.e the maximum temperature starts from index value 46 and ends at index value 53.

----loading and parsing data-----

```
A = load '/weatherPIG' using TextLoader as (data:chararray);

AF = foreach A generate TRIM(SUBSTRING(data, 6, 14)), TRIM(SUBSTRING(data, 46, 53)),
TRIM(SUBSTRING(data, 38, 45));

store AF into '/data9' using PigStorage(',');
```

```
S = load '/data9/part-m-00000' using PigStorage(',') as (date:chararray, min:double, max:double);

-------Hot Days------

X = filter S by max > 25;     /*  The days with temperature more than 25 are getting stored in the relation x */

-------Cold Days------

X = filter S by min < 0;   /* The days with temperature less than 0 is filtered and stored in the relation X */

-------Hottest Day-----

H1 = group S all;            /*To bring all the rows having cloumns named as date, minimum temperature and maximum temperature*/

I = foreach H1 generate MAX(S.max) as maximum; /* To find the maximum temperature from the rows stored in H1 */

X = filter S by max == I.maximum; /* To display the whole row containing the maximum temperature */

-------Coldest Day------

H2 = group S all;  /* To bring all the rows having cloumns named as date,minimum temperature and maximum temperature*/
```

J = foreach H2 generate MIN(S.min) as minimum; /* To find the minimum temperature from the rows stored in H1 */

X = filter S by min == J.minimum; /* To display the whole row containing the maximum temperature */

-----UDF-----

Step 1: register PIGUdfCorrupt.jar;

Step 2: A = load '/weatherPIG' using TextLoader as (data:chararray);

Step 3: AF = foreach A generate TRIM(SUBSTRING(data, 6, 14)), IfCorrupted(TRIM(SUBSTRING(data, 46, 53))), IfCorrupted(TRIM(SUBSTRING(data, 38, 45)));

Step 4: store AF into '/data2' using PigStorage(',');

Step 5: S = load '/data2/part-m-00000' using PigStorage(',') as (date:chararray, min:double, max:double);

Explanation for the above pig snippets:

Step 1: To add the external jars in the pig enviroment, we use the keyword register.

Step 2: The dataset weatherPIG has to be moved into hdfs and then We need to load it from HDFS in to the relation A.

When we load the dataset using TextLoader, we get resulting tuple all with single field with one line of input field.

Step 3: In this step we are extracting three columns i.e date, minimum temperature and maximum temperature based on the range of index given in the code snippet above.

Step 4: Store operator stores the result into HDFS. In this step we have stored the result in data2 directory

Step 5: On storing the relation to the directory present in hdfs, a part file is automatically created in the that directory, and we load that entire path into a new relation named as S

Usage of Store command using a sample example

1) Load the input file named as **testfile** from hdfs into relation x11.

```
grunt> x11 = Load '/testfile' using PigStorage(',') as (a1:int,a2:int,a3:int);
grunt> dump x11;
2014-07-25 14:56:41,203 [main] INFO  org.apache.pig.tools.pigstats.ScriptState -
 Pig features used in the script: UNKNOWN
```

2) Dump the relation x11 to see the contents.

```
grunt> dump x11;
2014-07-25 14:56:41,203 [main] INFO  org.apache.pig.tools.pigstats.ScriptState -
 Pig features used in the script: UNKNOWN
2014-07-25 14:56:41,203 [main] INFO  org.apache.pig.backend.hadoop.executionengi
ne.HExecutionEngine - pig.usenewlogicalplan is set to true. New logical plan wil
l be used.
2014-07-25 14:56:41,362 [main] INFO  org.apache.pig.backend.hadoop.executionengi
ne.HExecutionEngine - (Name: x11: Store(hdfs://localhost/tmp/temp-735751014/tmp1
843805093:org.apache.pig.impl.io.InterStorage) - scope-11 Operator Key: scope-11
```

```
ne.util.MapRedUtil - Total input paths to process : 1
(1,2,3)
(4,5,6)
```

3) Store the relation x11 into the directory named as **neloc** present inside hdfs.

```
grunt> store x11 into '/neloc' using PigStorage(',');
2014-07-25 14:57:35,118 [main] INFO  org.apache.pig.tools.pigstats.ScriptState -
 Pig features used in the script: UNKNOWN
2014-07-25 14:57:35,118 [main] INFO  org.apache.pig.backend.hadoop.executionengi
ne.HExecutionEngine - pig.usenewlogicalplan is set to true. New logical plan wil
l be used.
```

```
Output(s):
Successfully stored 2 records (12 bytes) in: "/neloc"

Counters:
Total records written : 2
Total bytes written : 12
Spillable Memory Manager spill count : 0
Total bags proactively spilled: 0
Total records proactively spilled: 0
```

4) See the content present inside the directory neloc by the below command.

   Command: hadoop dfs –ls /neloc

```
cloudera@cloudera-vm:~$ hadoop dfs -ls /neloc
Found 2 items
drwxr-xr-x   - cloudera supergroup          0 2014-07-25 14:57 /neloc/_logs
-rw-r--r--   1 cloudera supergroup         12 2014-07-25 14:57 /neloc/part-m-000
00
cloudera@cloudera-vm:~$ hadoop dfs -cat /neloc/part-m-00000
1,2,3
4,5,6
cloudera@cloudera-vm:~$
```