Pig Script examples:

Consider the student data file (st.txt) Data in the following format Name,

District, Age, gender

st.txt:

vamsi,prakasam,21,male

ramu,guntur,20,male

kalyani,prakasam,24,female

rajesh,prakasam,26,male

saraswathi,guntur,45,female

teekshita,guntur,5

**(i) Write a PIG Script to display Names of all Male Students.**

Ans:

grunt> rl = LOAD 'hdfs://localhost:9000/ex/st.txt' using PigStorage (',') as (Name: chararray, dist: chararray, age: int, gender: chararray);

grunt> dump rl;

**output:**

(vamsi,prakasam,21,male)

(ramu,guntur,20,male)

(kalyani,prakasam,24,female)

(rajesh,prakasam,26,male)

(saraswathi,guntur,45,female)

(teekshita,guntur,5,)

grunt> ml = FILTER rl by gender=='male';

grunt> dump ml;

**Output:**

(vamsi,prakasam,21,male)

(ramu,guntur,20,male)

(rajesh,prakasam,26,male)

grunt>fe = FOREACH ml GENERATE Name;

**output:**

(vamsi)

(ramu)

(rajesh)

**(ii) Write a PIG Script to find the number of students from Guntur**

**district.**

grunt> dd = FILTER rl by dist=='guntur';

grunt> dump dd;

**output:**

(ramu,guntur,20,male)

(saraswathi,guntur,45,female)

(teekshita,guntur,5,)

grunt> fe2 = FOREACH dd GENERATE Name;

grunt> dump fe2;

**output:**

(ramu)

(saraswathi)

(teekshita)

grunt>g = GROUP fe2 ALL;

grunt>dump g;

**Output:**

(all,{(teekshita),(saraswathi),(ramu)})

grunt>a\_count = FOREACH g GENERATE COUNT(fe2);

grunt>dump a\_count;

**output:**

(3)

**(iii) Write a PIG Script to display district wise count of all female students.**

grunt> fml = FILTER rl by gender=='female';

grunt> dump fml;

**Output:**

(kalyani,prakasam,24,female)

(saraswathi,guntur,45,female)

(teekshita,guntur,5,female)

grunt> gr = GROUP rl by dist;

grunt> dump gr;

**output:**

(guntur,{(teekshita,guntur,5,female),(saraswathi,guntur,45,female)})

(prakasam,{(kalyani,prakasam,24,female)})

grunt>co = FOREACH gr GENERATE group,Count(fml);

grunt>DUMP co;

**Output:**

(guntur,2)

(prakasam,1)

Relational Operators:

Relational operators are the main tools Pig Latin provides to operate on the data. It allows you to transform the data by sorting, grouping, joining, projecting and filtering. This section covers the basic relational operators.

LOAD:

LOAD operator is used to load data from the file system or HDFS storage into a Pig relation.

In this example, the Load operator loads data from file ‘first’ to form relation ‘loading1’. The field names are user, url, id.

FOREACH:

This operator generates data transformations based on columns of data. It is used to add or remove fields from a relation. Use FOREACH-GENERATE operation to work with columns of data.

FILTER:

This operator selects tuples from a relation based on a condition.

In this example, we are filtering the record from ‘loading1’ when the condition ‘id’ is greater than 8.

JOIN:

JOIN operator is used to perform an inner, equijoin join of two or more relations based on common field values. The JOIN operator always performs an inner join. Inner joins ignore null keys, so it makes sense to filter them out before the join.

In this example, join the two relations based on the column ‘url’ from ‘loading1’ and ‘loading2’.

ORDER BY:

Order By is used to sort a relation based on one or more fields. You can do sorting in ascending or descending order using ASC and DESC keywords.

DISTINCT:

Distinct removes duplicate tuples in a relation.Lets take an input file as below, which has amr,crap,8 and amr,myblog,10 twice in the file. When we apply distinct on the data in this file, duplicate entries are removed.

STORE:

Store is used to save results to the file system.

GROUP:The GROUP operator groups together the tuples with the same group key (key field). The key field will be a tuple if the group key has more than one field, otherwise it will be the same type as that of the group key. The result of a GROUP operation is a relation that includes one tuple per group.

COGROUP:

COGROUP is same as GROUP operator. For readability, programmers usually use GROUP when only one relation is involved and COGROUP when multiple relations re involved.

CROSS:

The CROSS operator is used to compute the cross product (Cartesian product) of two or more relations.

LIMIT:

LIMIT operator is used to limit the number of output tuples. If the specified number of output tuples is equal to or exceeds the number of tuples in the relation, the output will include all tuples in the relation.

SPLIT:

SPLIT operator is used to partition the contents of a relation into two or more relations based on some expression. Depending on the conditions stated in the expression.

**Apache Pig Grunt Shell Commands**

In order to write Pig Latin scripts, we use the Grunt shell of Apache Pig. By using sh and fs we can invoke any shell commands, before that.

i. sh Command

we can invoke any shell commands from the Grunt shell, using the sh command. But make sure, we cannot execute the commands that are a part of the shell environment (ex − cd), using the sh command.

Syntax

The syntax of the sh command is:

grunt> sh shell command parameters

Example

By using the sh option, we can invoke the ls command of Linux shell from the Grunt shell. Here, it lists out the files in the /pig/bin/ directory.

grunt> sh ls

pig

pig\_1444799121955.log

pig.cmd

pig.py

ii. fs Command

Moreover, we can invoke any fs Shell commands from the Grunt shell by using the fs command.

Syntax

The syntax of fs command is:

grunt> sh File System command parameters

Example

By using fs command, we can invoke the ls command of HDFS from the Grunt shell. Here, it lists the files in the HDFS root directory.

grunt> fs –ls

Found 3 items

drwxrwxrwx - Hadoop supergroup 0 2015-09-08 14:13 Hbase

drwxr-xr-x - Hadoop supergroup 0 2015-09-09 14:52 seqgen\_data

drwxr-xr-x - Hadoop supergroup 0 2015-09-08 11:30 twitter\_data

Similarly, using the fs command we can invoke all the other file system shell commands from the Grunt shell.

4. Utility Commands

It offers a set of Pig Grunt Shell utility commands. It involves clear, help, history, quiet, and set. Also, there are some commands to control Pig from the Grunt shell, such as exec, kill, and run. Here is the description of the utility commands offered by the Grunt shell.

i. clear Command

In order to clear the screen of the Grunt shell, we use Clear Command.

Syntax

The syntax of the clear command is:

grunt> clear

ii. help Command

The help command gives you a list of Pig commands or Pig properties.

Syntax

By using the help command, we can get a list of Pig commands.

grunt> help

Commands

<pig latin statement>;

See the Pig Latin manual for details:

hadoop.apache.org/

File system commands:

fs <fs arguments>

Equivalent to Hadoop dfs command:

http://hadoop.apache.org/common/docs/current/hdfs\_shell.html

Diagnostic Commands

describe <alias>[::<alias]

Show the schema for the alias.

Inner aliases can be described as A:: B.

[-script <pigscript>] [-out <path>] [-brief] [-dot|-xml]

[-param <param\_name>=<pCram\_value>]

[-param\_file <file\_name>] [<alias>] -

Show the execution plan to compute the alias or for the entire script.

-script: Explain the entire script.

-out: Store the output into directory rather than print to stdout.

-brief: Don’t expand nested plans (presenting a smaller graph for the overview).

-dot: Generate the output in .dot format. Default is text format.

-xml: Generate the output in .xml format. Default is text format.

-param <param\_name: See parameter substitution for details.

-param\_file <file\_name>: See parameter substitution for details.

alias: Alias to explain.

dump <alias>: Compute the alias and writes the results to stdout.

**Utility Commands**

exec [-param <param\_name>=param\_value] [-param\_file <file\_name>] <script> -

Execute the script with access to grunt environment including aliases.

-param <param\_name – See parameter substitution for details.

-param\_file <file\_name> – See parameter substitution for details.

script – Script to be executed.

run [-param <param\_name>=param\_value] [-param\_file <file\_name>] <script> -

Execute the script with access to grunt environment.

-param <param\_name: See parameter substitution for details.

-param\_file <file\_name>: See parameter substitution for details.

script: Script to be executed.

sh <shell command>: Invoke a shell command.

kill <job\_id>: Kill the hadoop job specified by the hadoop job id.

set <key> <value>: Provide execution parameters to Pig. Keys and values are case sensitive.

The following keys are supported:

default\_parallel: Script-level reduces parallelism. Basic input size heuristics used by default.

debug: Set debug on or off. The default is off.

job.name: A single-quoted name for jobs. Default is PigLatin:<script name>

job.priority: Priority for jobs. Values: very\_low, low, normal, high, very\_high.

Default is normal stream.skippath: String that contains the path.

This is used by streaming any Hadoop property.

help – Display this message.

history [-n] – Display the list statements in the cache.

-n – Hide line numbers.

quit – Quit the grunt shell.

iii. history Command

It is the very useful command, it displays a list of statements executed/used so far since the Grunt sell is invoked.

Syntax

Since opening the Grunt shell, let’s suppose we have executed three statements:

grunt> customers = LOAD 'hdfs://localhost:9000/pig\_data/customers.txt' USING PigStorage(',');

grunt> orders = LOAD 'hdfs://localhost:9000/pig\_data/orders.txt' USING PigStorage(',');

grunt> Employee = LOAD 'hdfs://localhost:9000/pig\_data/Employee.txt' USING PigStorage(',');

Then, using the history command will produce the following output.

grunt> history

customers = LOAD 'hdfs://localhost:9000/pig\_data/customers.txt' USING PigStorage(',');

orders = LOAD 'hdfs://localhost:9000/pig\_data/orders.txt' USING PigStorage(',');

Employee = LOAD 'hdfs://localhost:9000/pig\_data/Employee.txt' USING PigStorage(',');

iv. set Command

Basically, to show/assign values to keys, we use set command in Pig.

Syntax

There are several keys we can set values for, using this command. Such as:

default\_parallel

By passing any whole number as a value to this key, we can set the number of reducers for a map job.

debug

Also, by passing on/off to this key, we can turn off or turn on the debugging feature in Pig.

job.name

Moreover, by passing a string value to this key we can set the Job name to the required job.

job.priority

By passing one of the following values to this key, we can set the job priority to a job −

very\_low

low

normal

high

very\_high

stream.skippath

By passing the desired path in the form of a string to this key, we can set the path from where the data is not to be transferred, for streaming.

v. quit Command

We can quit from the Grunt shell, Using this command.

Syntax

It Quit from the Grunt shell:

grunt> quit

Now see the following commands. By using them we can control Apache Pig from the Grunt shell.

vi. exec Command

Using the exec command, we can execute Pig scripts from the Grunt shell.

Syntax

The syntax of the utility command exec is:

grunt> exec [–param param\_name = param\_value] [–param\_file file\_name] [script]

Example

Let’s suppose there is a file named Employee.txt in the /pig\_data/ directory of HDFS. Its content is:

Employee.txt

001,Mehul,Hyderabad

002,Ankur,Kolkata

003,Shubham,Delhi

Now, Suppose we have a script file named sample\_script.pig in the /pig\_data/ directory of HDFS. Its content is:

Sample\_script.pig

Employee = LOAD 'hdfs://localhost:9000/pig\_data/Employee.txt' USING PigStorage(',')

as (id:int,name:chararray,city:chararray);

Dump Employee;

Now, let us execute the above script from the Grunt shell using the exec command as shown below.

grunt> exec /sample\_script.pig

Output

The exec command executes the script in the sample\_script.pig. As directed in the script, it loads the Employee.txt file into Pig and gives you the result of the Dump operator displaying the following content.

(1,Mehul,Hyderabad)

(2,Ankur,Kolkata)

(3,Shubham,Delhi)

vii. kill Command

By using this command, we can kill a job from the Grunt shell.

Syntax

Given below is the syntax of the kill command.

grunt> kill JobId

Example

Assume there is a running Pig job having id Id\_0055. By using the kill command, we can kill it from the Grunt shell.

grunt> kill Id\_0055

viii. run Command

By using the run command, we can run a Pig script from the Grunt shell.

Syntax

The syntax of the run command is:

grunt> run [–param param\_name = param\_value] [–param\_file file\_name] script

Example

So, let’s suppose there is a file named Employee.txt in the /pig\_data/ directory of HDFS. Its content is:

Employee.txt

001,Mehul,Hyderabad

002,Ankur,Kolkata

003,Shubham,Delhi

Afterwards, suppose we have a script file named sample\_script.pig in the local filesystem. Its content is:

Sample\_script.pig

Employee= LOAD 'hdfs://localhost:9000/pig\_data/Employee.txt' USING

PigStorage(',') as (id:int,name:chararray,city:chararray);

Further, using the run command, let’s run the above script from the Grunt shell.

grunt> run /sample\_script.pig

Then, using the Dump operator, we can see the output of the script.

grunt> Dump;

(1,Mehul,Hyderabad)

(2,Ankur,Kolkata)

(3,Shubham,Delhi)

Also, it is very important to note that there is one difference between exec and the run command. That is if we use to run, the statements from the script are available in the command history.