

Hadoop Basics with InfoSphere BigInsights

Lesson 4: Querying data



Catalog Number

Contents

Lab 1Working with Jaql, Pig, and Hive.....	4
1.1Working with Jaql (Terminal).....	5
1.2Working with Jaql (Eclipse).....	9
1.3Working with Pig (Terminal).....	15
1.4Working with Pig (Eclipse).....	17
1.5Working with Hive.....	23
1.6Summary.....	25

Lab 1 Working with Jaql, Pig, and Hive

If you're looking to get off to a quick start with “big data” projects involving IBM's InfoSphere BigInsights, learning the basics of how to query, and manipulate your data is important. Working with big data often requires querying that data to isolate information of interest and manipulate it in various ways. This hands-on lab takes you through simple query examples that illustrate how you can read, write, and filter data.

This lab introduces you to Jaql, Pig, and Hive, query languages provided with InfoSphere BigInsights, and explores how you can query data.

After completing this hands-on lab, you'll be able to:

- Write and execute simple Jaql scripts, through terminal and Eclipse
- Write and execute simple Pig scripts, through terminal and Eclipse
- Write and execute simple Hive commands through terminal.

Allow 45 minutes to 1 hour complete this section of lab.

This version of the lab was designed using the InfoSphere BigInsights 2.1 Quick Start Edition. Throughout this lab you will be using the following account login information:

	Username	Password
VM image setup screen	root	password
Linux	biadmin	biadmin

If you are continuing this series of hands on labs immediately after completing Hadoop Basics Unit 1: Exploring Hadoop Distributed File System, you may move on to section 1.1 of this lab. Otherwise please refer to Hadoop Basics Unit 1: Exploring Hadoop Distributed File System Section 1.1 to get started. (All Hadoop components should be running)

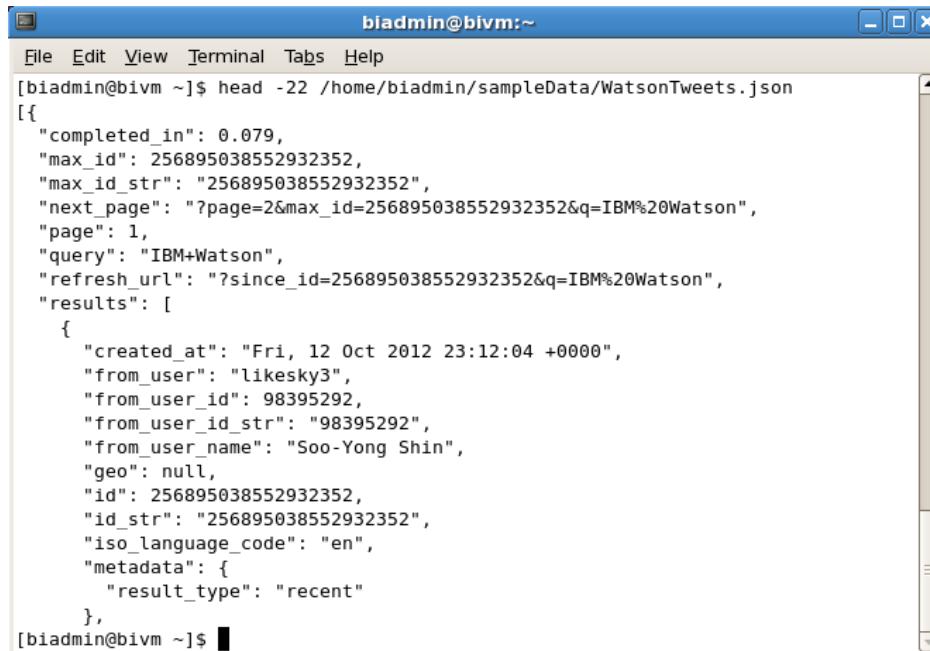
1.1 Working with Jaql (Terminal)

In this section we will be working with a JSON Array file called WatsonTweets.json.

- ___1. Let examine the format of the Watson Tweets. Open a terminal and execute the following commands.

```
head -22 /home/biadmin/sampleData/WatsonTweets.json
```

Your output should look similar to the image below.

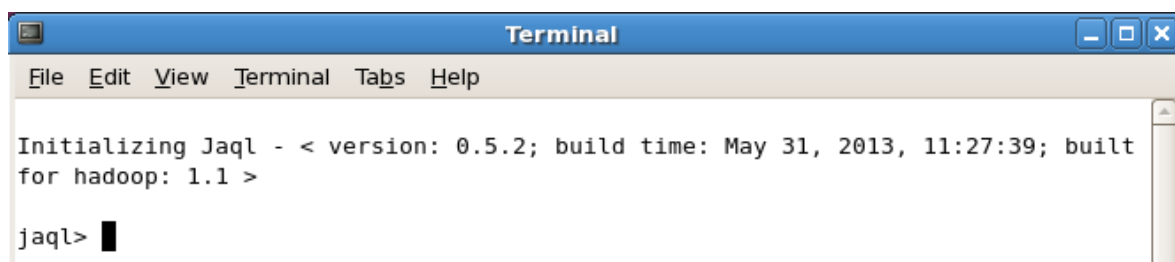


```
biadmin@bivm:~
File Edit View Terminal Tabs Help
[biadmin@bivm ~]$ head -22 /home/biadmin/sampleData/WatsonTweets.json
[{
  "completed_in": 0.079,
  "max_id": 256895038552932352,
  "max_id_str": "256895038552932352",
  "next_page": "?page=2&max_id=256895038552932352&q=IBM%20Watson",
  "page": 1,
  "query": "IBM+Watson",
  "refresh_url": "?since_id=256895038552932352&q=IBM%20Watson",
  "results": [
    {
      "created_at": "Fri, 12 Oct 2012 23:12:04 +0000",
      "from_user": "likesky3",
      "from_user_id": 98395292,
      "from_user_id_str": "98395292",
      "from_user_name": "Soo-Yong Shin",
      "geo": null,
      "id": 256895038552932352,
      "id_str": "256895038552932352",
      "iso_language_code": "en",
      "metadata": {
        "result_type": "recent"
      }
    }
  ]
}]
[biadmin@bivm ~]$
```

- ___2. You can open a Jaql terminal by clicking on the BigInsights Shell icon, then on the Jaql terminal icon.



The jaql shell should look like the image below.



```
Terminal
File Edit View Terminal Tabs Help

Initializing Jaql - < version: 0.5.2; build time: May 31, 2013, 11:27:39; built
for hadoop: 1.1 >

jaql>
```

- ___3. We want to extract some data from the *WatsonTweets.json* file and save it into a Sequence file. Specifically the data that is in the nested *results* array. Execute the following code:

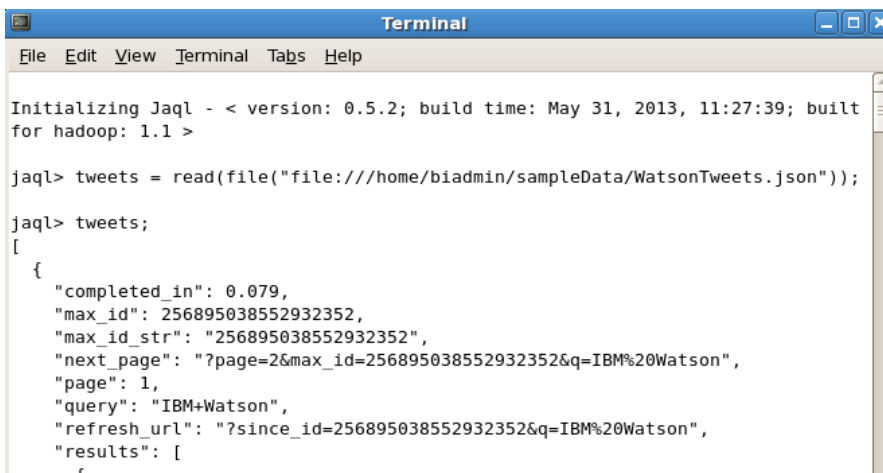
```
tweets = read(file("file:///home/biadmin/sampleData/WatsonTweets.json"))
);
```

This line of code reads the file from your local file system and stores the content in the *tweets* variable.

- ___4. To check whether the file was read type the following:

```
tweets;
```

This will execute the *tweets* statement and show the results. It should look similar to the image below.



```
Terminal
File Edit View Terminal Tabs Help

Initializing Jaql - < version: 0.5.2; build time: May 31, 2013, 11:27:39; built
for hadoop: 1.1 >

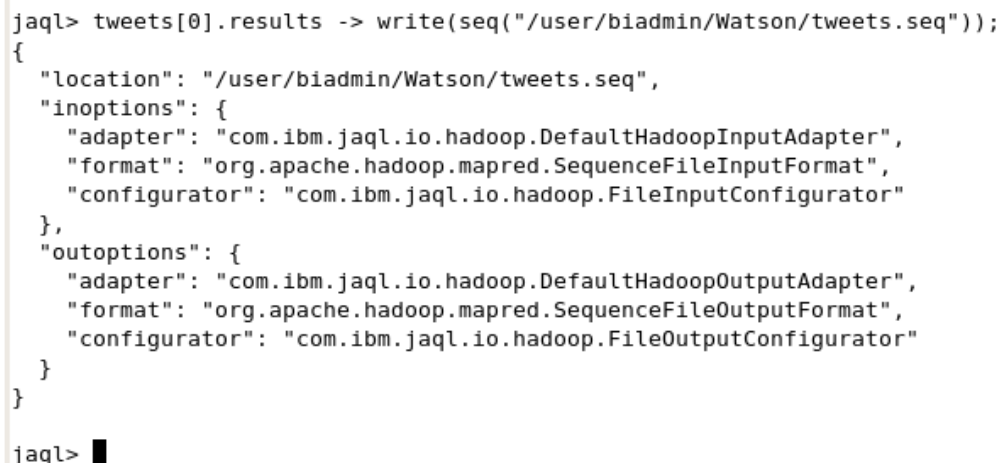
jaql> tweets = read(file("file:///home/biadmin/sampleData/WatsonTweets.json"));

jaql> tweets;
[
  {
    "completed_in": 0.079,
    "max_id": 256895038552932352,
    "max_id_str": "256895038552932352",
    "next_page": "?page=2&max_id=256895038552932352&q=IBM%20Watson",
    "page": 1,
    "query": "IBM+Watson",
    "refresh_url": "?since_id=256895038552932352&q=IBM%20Watson",
    "results": [
      {
```

Notice how there is a nested array called *results*. All we want from *WatsonTweets.json* file is that *results* nested array. We will isolate that array and save it in a sequence file back into HDFS where we may filter it.

- ___5. Execute the following code to extract the *results* array and save it to HDFS.

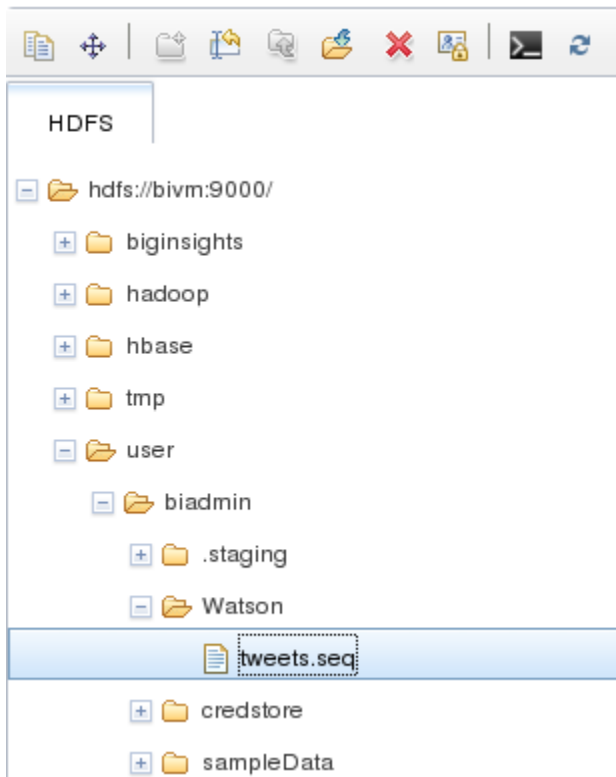
```
tweets[0].results -> write(seq("/user/biadmin/Watson/tweets.seq"));
```



```
jaql> tweets[0].results -> write(seq("/user/biadmin/Watson/tweets.seq"));
{
  "location": "/user/biadmin/Watson/tweets.seq",
  "inoptions": {
    "adapter": "com.ibm.jaql.io.hadoop.DefaultHadoopInputAdapter",
    "format": "org.apache.hadoop.mapred.SequenceFileInputFormat",
    "configurator": "com.ibm.jaql.io.hadoop.FileInputConfigurator"
  },
  "outoptions": {
    "adapter": "com.ibm.jaql.io.hadoop.DefaultHadoopOutputAdapter",
    "format": "org.apache.hadoop.mapred.SequenceFileOutputFormat",
    "configurator": "com.ibm.jaql.io.hadoop.FileOutputConfigurator"
  }
}

jaql>
```

You can check that the file was written to HDFS by going to the Web Console under the Files tab and navigating to the appropriate directory.



Now that we have some data to use in HDFS we can filter that data and get information out of it, such as the language the tweets were written in.

Before filtering we must first read that data into another variable.

__6. Save the newly written file into another variable.

```
tweetsHDFS = read(seq("/user/biadmin/Watson/tweets.seq"));
```

Now that we have a variable with just the *results* array (Remember that tweets.seq is the *results* array from the original WatsonTweets.json file) we can filter that data however we like.

__7. Issue the command below to filter the data by the language that they were written in.

```
tweetsHDFS -> transform $.iso_language_code;
```

You should get the output below.

```
jaql> tweetsHDFS = read(seq("/user/biadmin/Watson/tweets.seq"));  
  
jaql> tweetsHDFS -> transform $.iso_language_code;  
[  
  "en",  
  "en",  
  "en",  
  "en",  
  "en",  
  "en",  
  "es",  
  "pt",  
  "ja",  
  "en",  
  "en",  
  "en",  
  "es",  
  "en"  
]  
  
jaql> █
```

__8. To quit the jaqlshell simply type the following and hit enter:

```
quit;
```

You have now learned how to write jaql statements in terminal. The next step is to do a program similar to this one with the Eclipse tooling!

1.2 Working with Jaql (Eclipse)

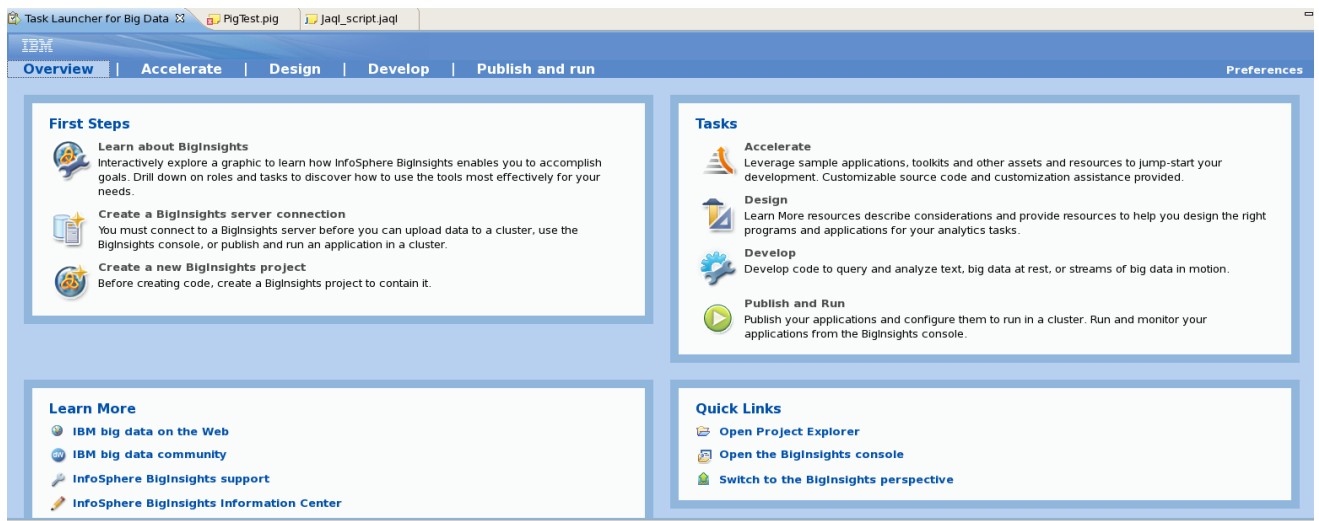
In this section we will learn how to write and run a jaql script within BigInsights Studio (Eclipse).

- __1. Open BigInsights Studio by clicking on the Eclipse icon.

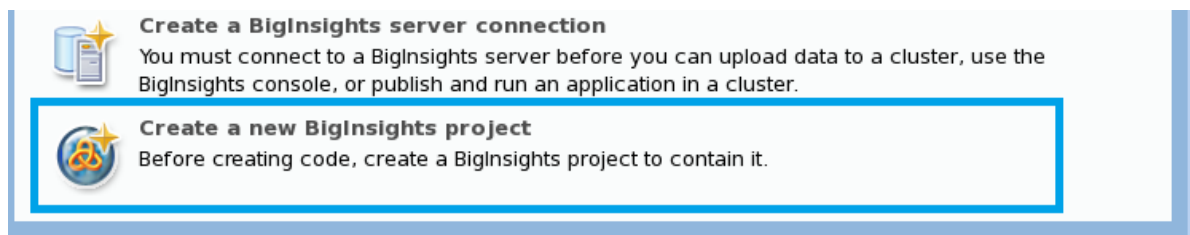


- __2. Click ok for the default workspace.

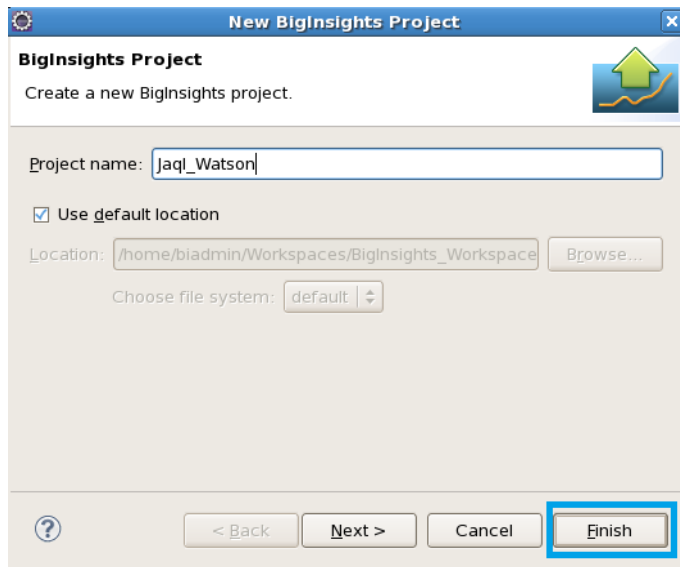
- __3. You should now be in the Overview tab which looks like the image below.



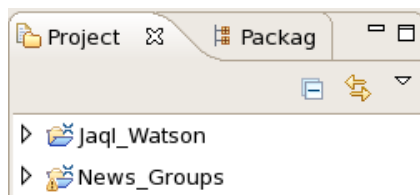
- __4. Click on “Create a new BigInsights project”



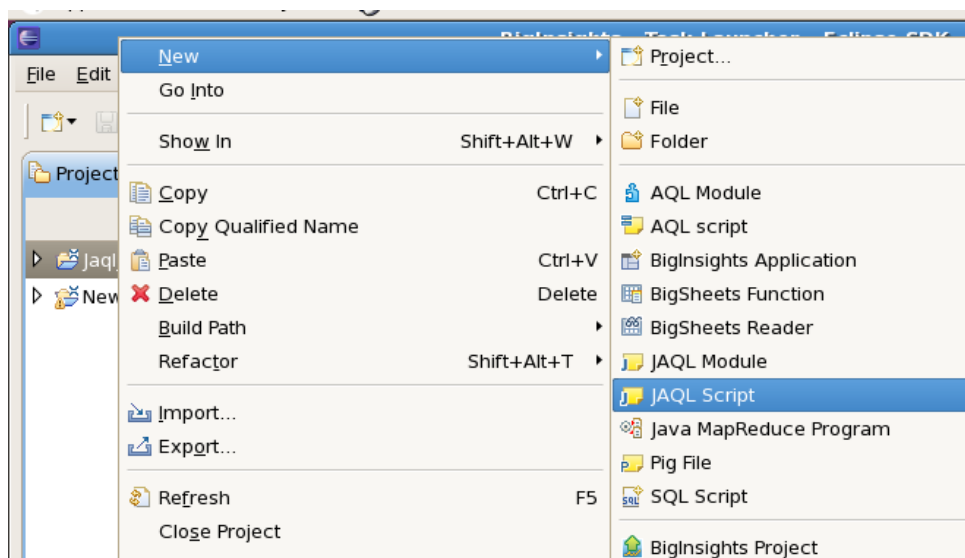
- ___5. A new project wizard will open requesting a project name. Name the project **Jaql_Watson**. Your wizard should look similar to the image below. Make sure the *Use Default Location* check box is checked. Then click *Finish*.



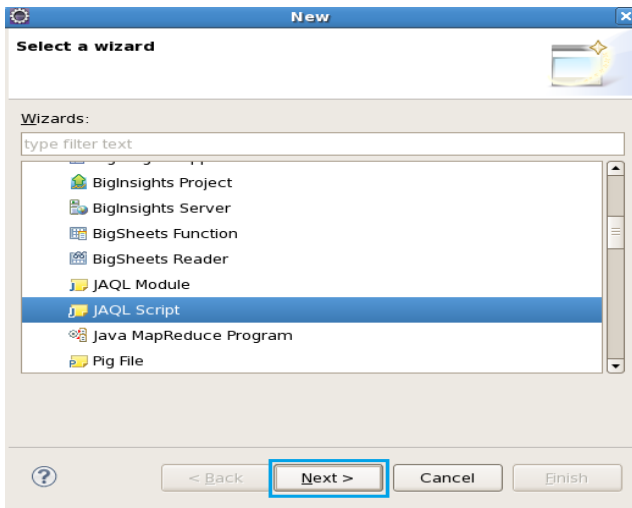
- ___6. Your new project should now appear in the Project Explorer pane on the left



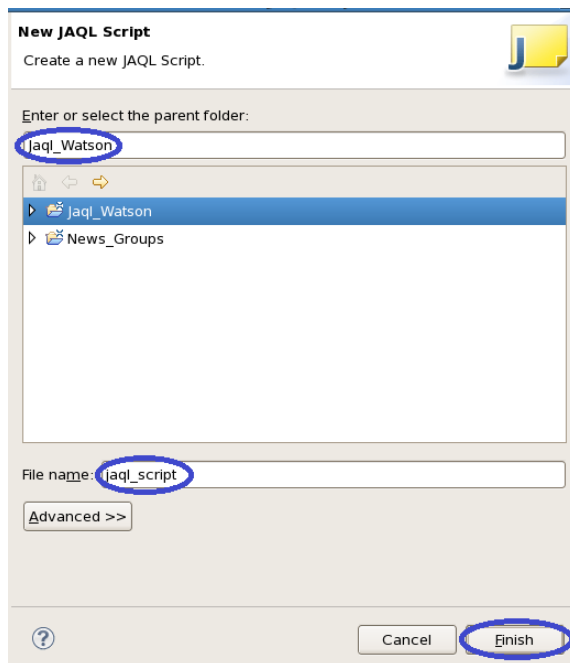
- ___7. Right-click the *Jaql_Watson* project, navigate to *New* then look for the option *JAQL Script* and select it.



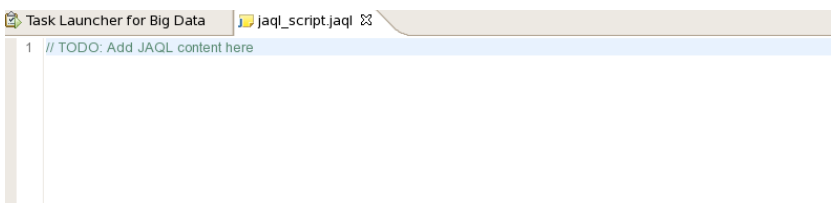
If JAQL Script is not visible as an option, then click on *Other*. A new window will appear, select JAQL Script then click *Next*.



- __8. A new JAQL Script wizard will appear, select *Jaql_Watson* as the parent folder, then name the file **jaql_script**. Click *Finish*.



A page similar to the one below will appear. This is where you will write your jaql statements.



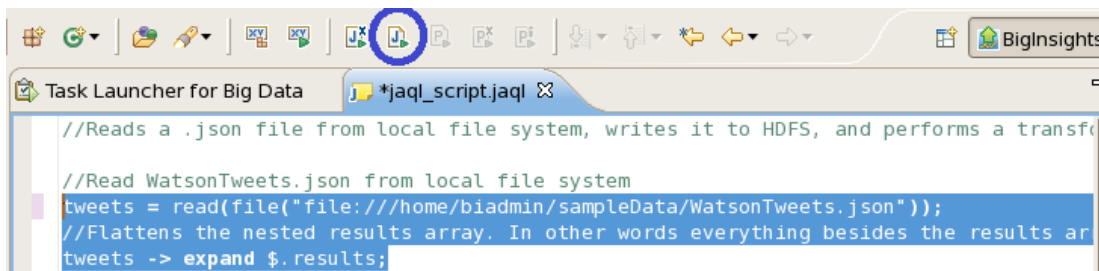
__9. Enter the following code:

```
//Reads a .json file from local file system, writes it to HDFS, and performs a
transform on the data.
//Read WatsonTweets.json from local file system
tweets = read(file("file:///home/biadmin/sampleData/WatsonTweets.json"));
//Flattens the nested results array. In other words everything besides the results
array is removed.
tweets -> expand $.results;
```

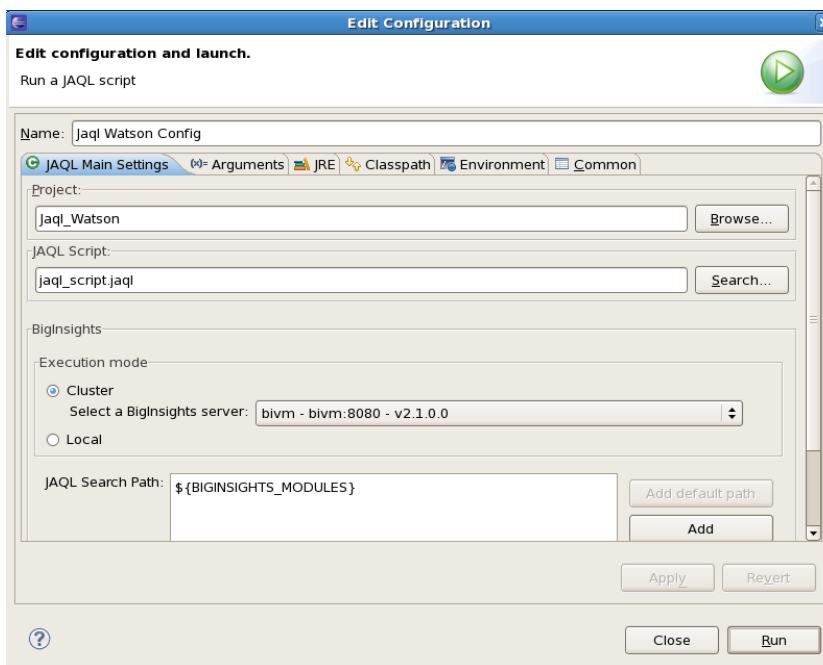
Save the file. Now to run this script you will need to high light the code segment that you wish to run and click on the *Run the JAQL Statement* icon.



__10. Highlight the two jaql statements and click the *Run the JAQL Statement*.



A new run configuration will appear. You may change the name to anything you wish. In this case it was named *Jaql Watson Config*. Also make sure that the correct project and script is selected. For the Execution Mode select *Cluster* and choose your server. Your configuration should look similar the image below. Click Run



If you look back to Section 2, number 4, you can compare the output of running `tweets;` vs `tweets -> expand $.results;`

```
[MapReduce Jobs Left: 0; Estimated Work Left: 0.00%
{
  "created_at": "Fri, 12 Oct 2012 23:12:04 +0000",
  "from_user": "likesky3",
  "from_user_id": 98395292,
  "from_user_id_str": "98395292",
  "from_user_name": "Soo-Yong Shin",
  "geo": null,
  "id": 256895038552932352,
  "id_str": "256895038552932352",
  "iso_language_code": "en",
  "metadata": {
    "result_type": "recent"
  },
  "profile_image_url": "http://a0.twimg.com/profile_images/1462807106/_____normal.jpg",
  "profile_image_url_https": "https://si0.twimg.com/profile_images/1462807106/_____normal.jpg",
  "source": "&lt;a href=&quot;http://twicca.r246.jp/&quot;&gt;twicca&lt;/a&gt;",
  "text": "RT @westr: RT @IBMWatson: @HPinsider Here's an animation of how #ibmwatson answers c",
  "to_user": null,
  "to_user_id": 0,
  "to_user_id_str": "0",
  "to_user_name": null
},
```

The latter contains only the data from the *results* array.

__11. Now lets run the code from the previous section. Copy and paste the following code into your script:

```
//Reads a .json file from local file system, writes it to HDFS, and performs a
transform on the data.

//Read WatsonTweets.json from local file system
tweets = read(file("file:///home/biadmin/sampleData/WatsonTweets.json"));

//Flattens the nested array. In other words everything besides the results array is
removed.
//tweets -> expand $.results;

//Take the results nested array from WatsonTweets.json and write it to HDFS
tweets[0].results -> write(seq("/user/biadmin/Watson/tweetsE.seq"));

//Read the seq file that was just written to HDFS and extract the 'from_user_id' from
tweets
tweetsHDFS = read(seq("/user/biadmin/Watson/tweetsE.seq"));

//Extraction of the 'from_user_id' from tweetsE.seq file
tweetsHDFS -> transform $.from_user_id;
```

- ___12. Highlight all of the statements and click *Run the JAQL Statements*. Notice how the *tweets* -> *expand \$.results;* was commented out. This is because we don't want to clutter the output screen. Your output should

```
MapReduce Jobs Left: 0; Estimated Work Left: 0.00%
{
  "inoptions": {
    "adapter": "com.ibm.jaql.io.hadoop.DefaultHadoopInputAdapter",
    "configurator": "com.ibm.jaql.io.hadoop.FileInputConfigurator",
    "format": "org.apache.hadoop.mapred.SequenceFileInputFormat"
  },
  "location": "/user/biadmin/Watson/tweetsE.seq",
  "outoptions": {
    "adapter": "com.ibm.jaql.io.hadoop.DefaultHadoopOutputAdapter",
    "configurator": "com.ibm.jaql.io.hadoop.FileOutputConfigurator",
    "format": "org.apache.hadoop.mapred.SequenceFileOutputFormat"
  }
}
MapReduce Jobs Left: 1; Estimated Work Left: 100.00%
MapReduce Jobs Left: 1; Estimated Work Left: 50.00%; Active Jobs: job_201305221047_0002.jaql job
MapReduce Jobs Left: 1; Estimated Work Left: 0.00%; Active Jobs: job_201305221047_0002.jaql job
MapReduce Jobs Left: 0; Estimated Work Left: 0.00%
[
  199737585,
  17873566,
  14267832,
  96648530,
  874511936,
  98395292,
  466711376,
  784932223,
  234352293,
  20661208,
  226793352,
  17466386,
  283439211,
  432651637
]
```

All user id's have now been extracted from the WatsonTweets.json file.

You have now learned how to write a simple jaql script in Eclipse.

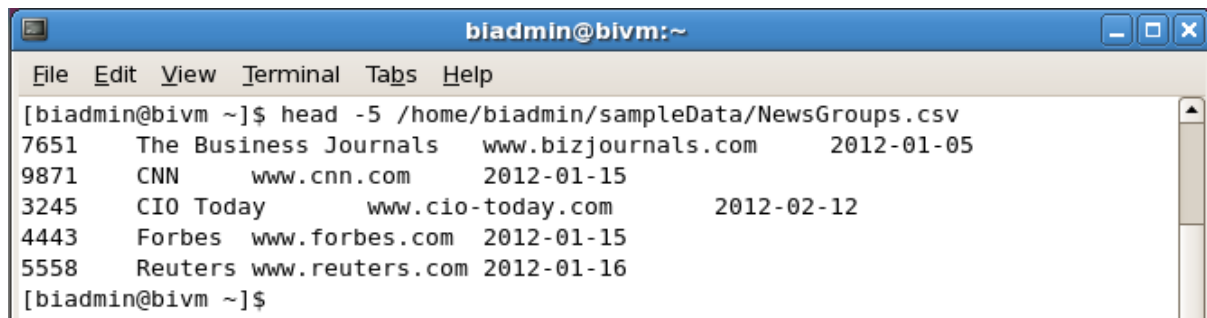
1.3 Working with Pig (Terminal)

Now that we have experienced JAQL through terminal and the Eclipse tooling, it is time to learn PIG. In this section we are going to use apache Pig to process a fictional data file called NewsGroups.csv. This file contains some newspaper names and some contact dates.

Let us examine the format of the NewsGroups.csv file.

- __1. Execute the commands below to examine the format of the records:

```
head -5 /home/biadmin/sampleData/NewsGroups.csv
```



```

biadmin@bivm:~
File Edit View Terminal Tabs Help
[biadmin@bivm ~]$ head -5 /home/biadmin/sampleData/NewsGroups.csv
7651 The Business Journals www.bizjournals.com 2012-01-05
9871 CNN www.cnn.com 2012-01-15
3245 CIO Today www.cio-today.com 2012-02-12
4443 Forbes www.forbes.com 2012-01-15
5558 Reuters www.reuters.com 2012-01-16
[biadmin@bivm ~]$

```

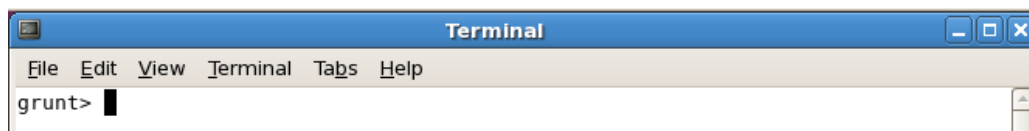
- __2. Copy the data into HDFS in case it is not there already:

```
hadoop fs -put /home/biadmin/sampleData/NewsGroups.csv
/user/biadmin/sampleData/NewsGroups.csv
```

- __3. You can open a Pig terminal by clicking the BigInsights Shell icon, then clicking on Pig



Your terminal should look similar to the image below.



```

Terminal
File Edit View Terminal Tabs Help
grunt>

```

NOTE: If your pig terminal does not run, ensure BigInsights has been started.

- __4. The first step in processing the data is to LOAD it. Execute the step below to load data.

```
records = LOAD 'sampleData/NewsGroups.csv' AS (id: int, name: chararray, url:
chararray, date: int);
```

This returns instantly. The processing is delayed until the data needs to be reported.

__5. To produce a histogram, we want to group by ID numbers.

```
grouped = GROUP records by id;
```

__6. Count the number of times each newsgroup was contacted based on the number of IDs that repeat using FOREACH GENERATE command.

```
final = FOREACH grouped GENERATE group, COUNT(records.id);
```

__7. Use the DUMP command to print the result to the console. This will cause all the previous steps to be executed.

```
DUMP final;
```

This should produce output similar to the following image:

```
Terminal
File Edit View Terminal Tabs Help
grunt> records = LOAD '/user/biadmin/sampleData/NewsGroups.csv' AS (id: int, name: chararray, url: chararray, date: int);
grunt> grouped = GROUP records by id;
grunt> final = FOREACH grouped GENERATE group, COUNT(records.id);
grunt> DUMP final;
INFO [Thread-18] org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
WARN [main] org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Encountered Warning FIELD_DISCARDED_TYPE_CONVERSION_FAILED 18 time(s).
INFO [main] org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
(3245,2)
(4443,2)
(5558,2)
(6543,2)
(7651,1)
(7652,2)
(7776,2)
(8761,2)
(9871,3)
grunt>
```

__8. Quit pig.

```
quit;
```

You have now learned how to write pig statements in terminal. The next step is to write a program similar to this one with the Eclipse tooling!

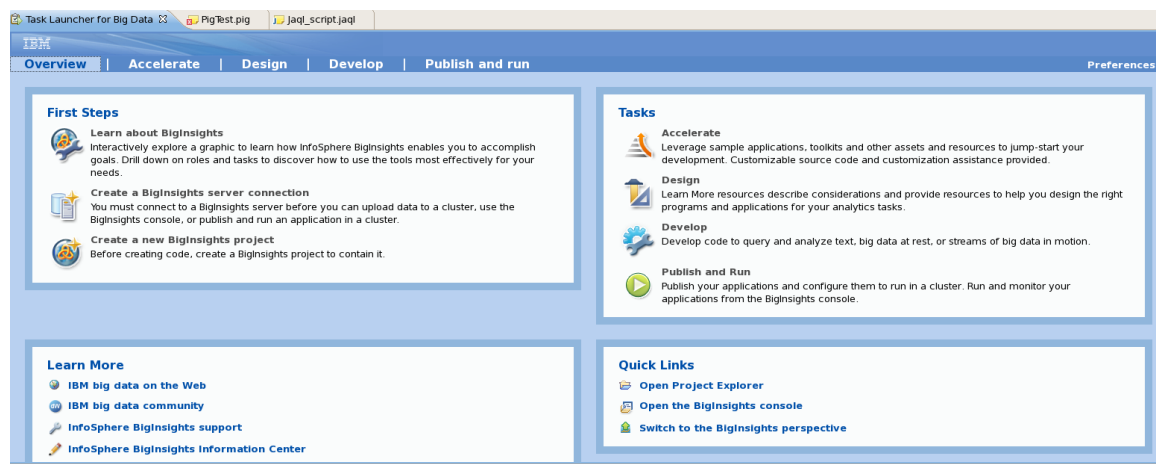
1.4 Working with Pig (Eclipse)

In this section we will learn how to write and run a pig script within Eclipse.

- __1. If Eclipse is not already open double-click on the Eclipse icon.



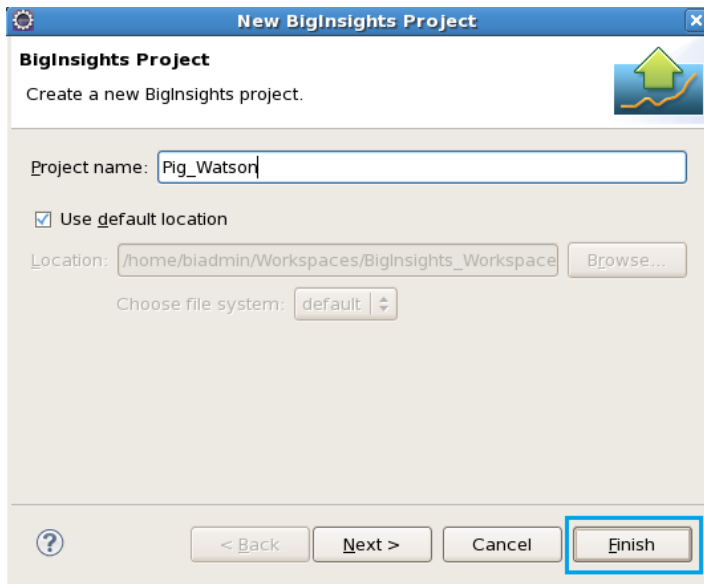
- __2. If prompted for a workspace, select the default and click OK.
- __3. You should now be in the Overview tab, which looks similar to the image below.



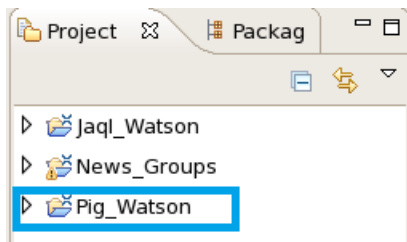
- __4. Now click on **Create a new BigInsights project**



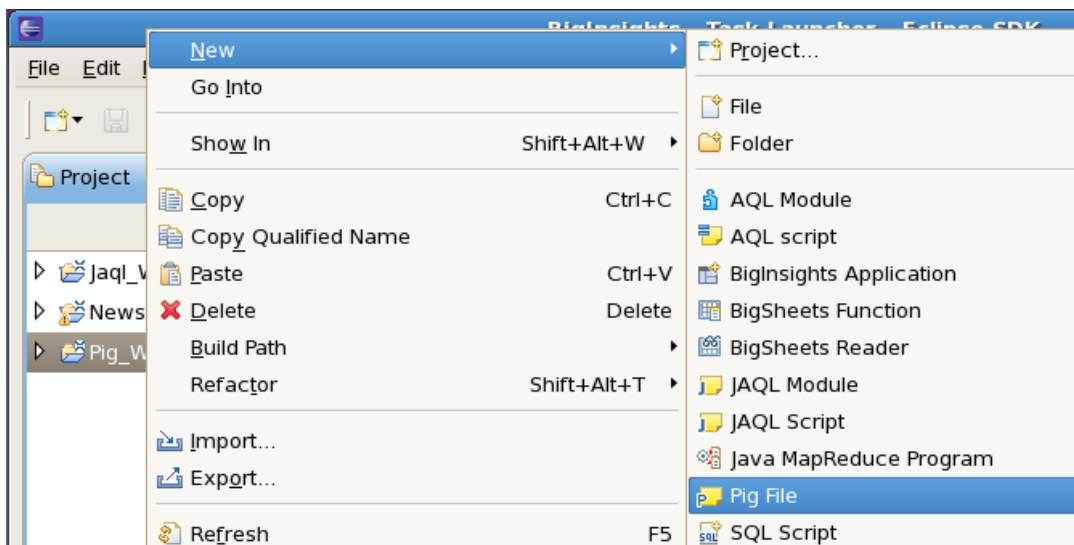
- __5. A new project wizard will open requesting a project name. Name the project **Pig_Watson**. Your wizard should look similar to the image below. Make sure the *Use Default Location* check box is checked. Then click *Finish*



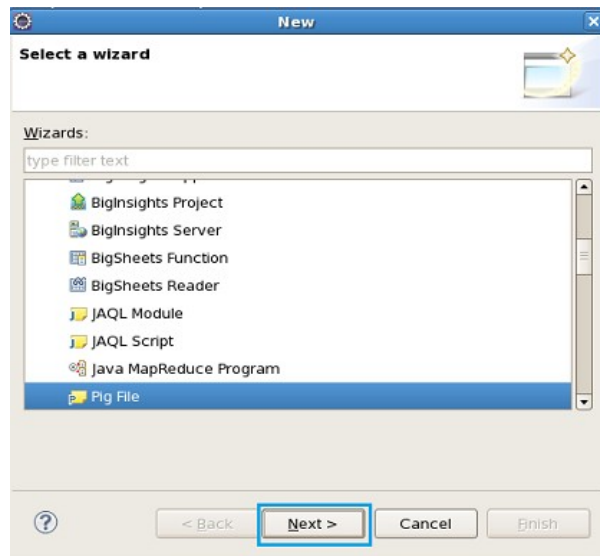
- __6. Your new project should now appear in the *Project Explorer* pane on the left.



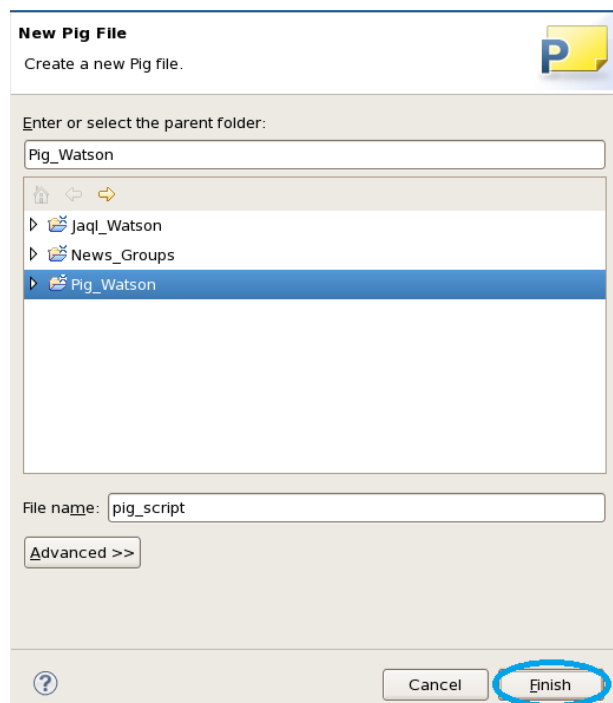
- __7. Right-click the *Pig_Watson* project, navigate to *New* then look for the option *Pig File* and click it.



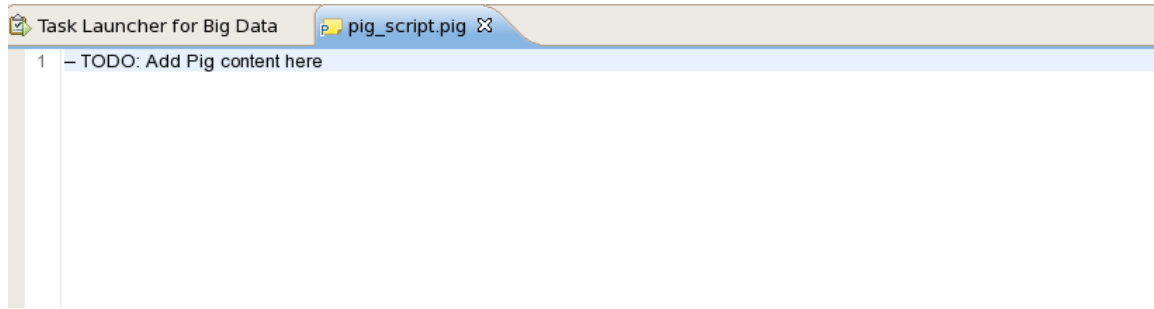
If Pig File is not visible as an option, then click on *Other*. A new window will appear, select Pig File then click *Next*.



__8. A new Pig File wizard will appear, select *Pig_Watson* as the parent folder, then name the file **pig_script**. Click *Finish*.



A page similar to the one below will appear. This is where you will write your pig statements.



__9. Enter the following code:

-- TODO: Add Pig content here

```
-- Load the data from HDFS with the given schema
records = LOAD '/user/biadmin/sampleData/NewsGroups.csv' AS (id: int, name:
chararray, url: chararray, date: int);
```

```
-- Arrange the records by IDs
grouped = group records by id;
```

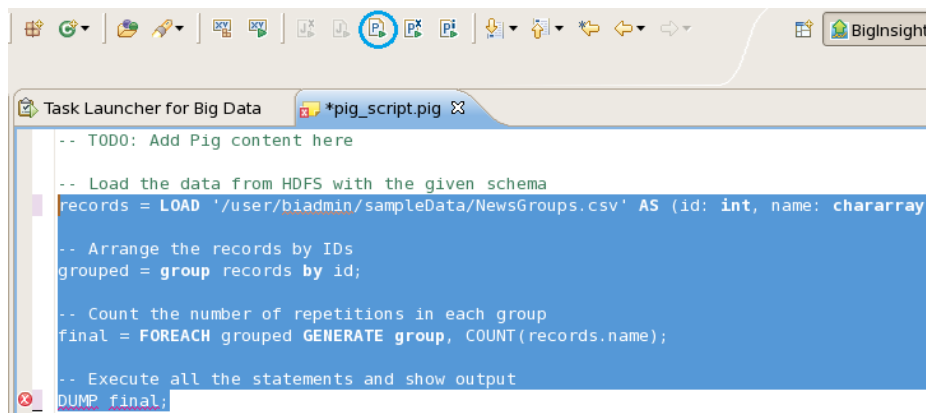
```
-- Count the number of repetitions in each group
final = FOREACH grouped GENERATE group, COUNT(records.name);
```

```
-- Execute all the statements and show output
```

```
DUMP final;
```

Hit Ctrl+S to save. Now to run this script you will need to highlight the code segment that you wish to run and click on the *Run Pig Statements icon*. Ignore the error on line 13.

__10. Highlight all the pig statements like shown in the image below and click the *Run Pig Statements icon*. Ignore the error on line 13.



A new run configuration will appear. You may change the name to anything you wish. In this case it was named *Pig Watson Config*. Also make sure that the correct project and script is selected. For the Execution Mode select *Cluster* and choose your server. Your configuration should look similar the image below.



NOTE: If you have not created a server yet please check back to Unit 2, Section 4.6

Edit configuration and launch.

Run a Pig script

Name: pig_script.pig

Pig Main Settings Arguments JRE Classpath Environment Common

Project: Pig_Watson

Pig Script: pig_script.pig

BigInsights

Execution mode

☒ Cluster

Select a BigInsights server: localhost - localhost:8080 - v2.1.0.0

☐ Local

Parameter Name	Parameter Value

Apply Revert

Run Close

Click Apply then Run.

The output should look similar to the image below.

```
Job DAG:
job_201305221047_0024

2013-05-23 09:28:08,021 [main] WARN org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Encounte
2013-05-23 09:28:08,021 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2013-05-23 09:28:08,027 [main] INFO org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2013-05-23 09:28:08,027 [main] INFO org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(3245,2)
(4443,2)
(5558,2)
(6543,2)
(7651,1)
(7652,2)
(7776,2)
(8761,2)
(9871,3)
```

___11. Now we want to store the output back to HDFS. Comment out the last statement with two hyphens (--), and replace add the following code:

```
-- Store output back into HDFS
STORE final INTO '/user/biadmin/sampleData/pig_output' USING
PigStorage();
```

Now highlight everything like in the image below and run the script.

```
-- TODO: Add Pig content here

-- Load the data from HDFS with the given schema
records = LOAD '/user/biadmin/sampleData/NewsGroups.csv' AS (id: int, name: chararray);

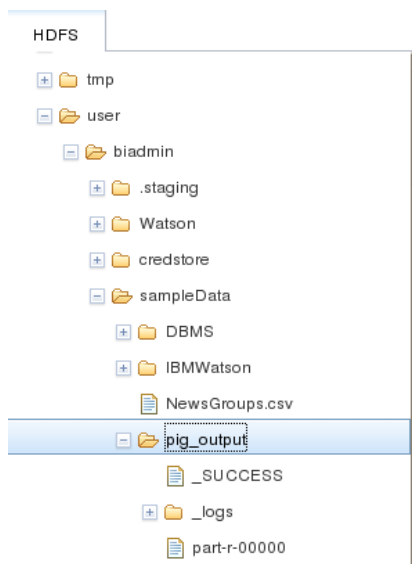
-- Arrange the records by IDs
grouped = group records by id;

-- Count the number of repetitions in each group
final = FOREACH grouped GENERATE group, COUNT(records.name);

-- Execute all the statements and show output
--DUMP final;

--Store output back into HDFS
STORE final INTO '/user/biadmin/sampleData/pig_output' USING PigStorage();
```

If everything is executed correctly you will have a new folder under the sampleData directory in HDFS.



Click the part-r-00000 file to see the output. It will look similar to the image below.

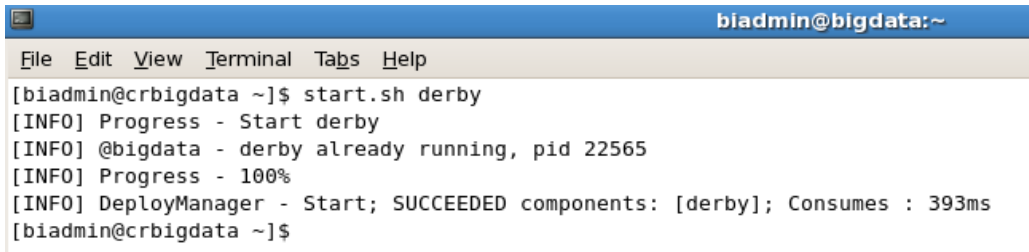
part-r-00000	63 B	128.0 MB
part-r-00000	63 B	128.0 MB
<div> <div>Edit</div> <div>Viewing Size: 10KB</div> <div>Text Sheet</div> </div>		
3245	2	
4443	2	
5558	2	
6543	2	
7651	1	
7652	2	
7776	2	
8761	2	
9871	3	

1.5 Working with Hive

Now that we have experienced JAQL and PIG through terminal and the Eclipse tooling, it is time to learn HIVE. In this section we are going to use apache Hive to process a data file called NewsGroups.csv. This file contains newspaper ids, names, urls, and some contact dates used only as samples.

- __12. Ensure the Apache Derby component is started. Apache Derby is the default database used as metastore in Hive. A quick way to verify if it is started, is to try to start using:

start.sh derby

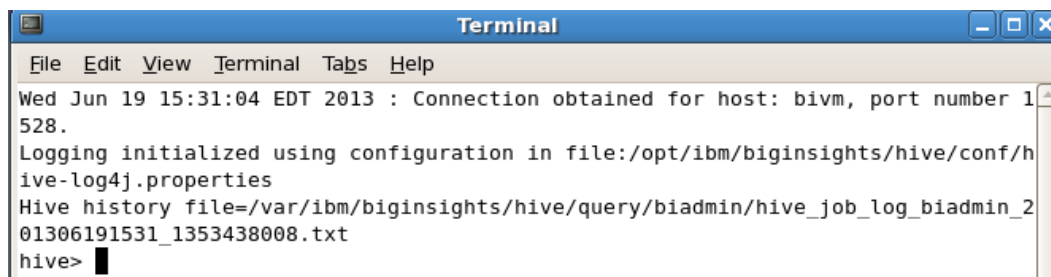


```

biadmin@bigdata:~
File Edit View Terminal Tabs Help
[biadmin@crbigdata ~]$ start.sh derby
[INFO] Progress - Start derby
[INFO] @bigdata - derby already running, pid 22565
[INFO] Progress - 100%
[INFO] DeployManager - Start; SUCCEEDED components: [derby]; Consumes : 393ms
[biadmin@crbigdata ~]$

```

- __13. You can start a Hive shell by clicking on the BigInsights Shell icon, then the Hive shell.

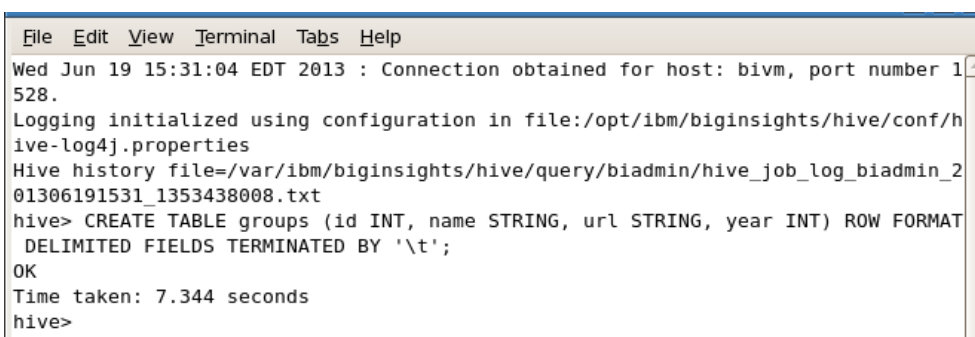
```

Terminal
File Edit View Terminal Tabs Help
Wed Jun 19 15:31:04 EDT 2013 : Connection obtained for host: b1vm, port number 1528.
Logging initialized using configuration in file:/opt/ibm/biginsights/hive/conf/hive-log4j.properties
Hive history file=/var/ibm/biginsights/hive/query/biadmin/hive_job_log_biadmin_201306191531_1353438008.txt
hive>

```

- __14. Create a table called groups.

CREATE TABLE groups (id INT, name STRING, url STRING, year INT) ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t';



```

File Edit View Terminal Tabs Help
Wed Jun 19 15:31:04 EDT 2013 : Connection obtained for host: b1vm, port number 1528.
Logging initialized using configuration in file:/opt/ibm/biginsights/hive/conf/hive-log4j.properties
Hive history file=/var/ibm/biginsights/hive/query/biadmin/hive_job_log_biadmin_201306191531_1353438008.txt
hive> CREATE TABLE groups (id INT, name STRING, url STRING, year INT) ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t';
OK
Time taken: 7.344 seconds
hive>

```

__15. Load the data from NewsGroups.csv into the groups table

```
LOAD DATA LOCAL INPATH '/home/biadmin/sampleData/NewsGroups.csv' OVERWRITE
INTO TABLE groups;
```

```
hive> LOAD DATA LOCAL INPATH '/home/biadmin/sampleData/NewsGroups.csv' OVERWRITE
INTO TABLE groups;
Copying data from file:/home/biadmin/sampleData/NewsGroups.csv
Copying file: file:/home/biadmin/sampleData/NewsGroups.csv
Loading data to table default.groups
Deleted hdfs://bivm:9000/biginsights/hive/warehouse/groups
OK
Time taken: 0.326 seconds
hive>
```

__16. Create a table named counts to store the counts for each newsgroup for our histogram.

```
CREATE TABLE counts (thenames STRING, theids INT);
```

```
OK
Time taken: 0.316 seconds
hive> CREATE TABLE counts (thenames STRING, theids INT);
OK
Time taken: 0.108 seconds
hive>
```

__17. Fill the counts table with the names and ids data from the groups table.

```
INSERT OVERWRITE TABLE counts SELECT name, id FROM groups;
```

```
hive> INSERT OVERWRITE TABLE counts SELECT name, id FROM groups;
Total MapReduce jobs = 2
Launching Job 1 out of 2
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_201305221047_0036, Tracking URL = http://bigdata:50030/jobdetails.jsp?jobid=job_201305221047_0036
Kill Command = /lib/BigInsights/biginsights/IHC/libexec/./bin/hadoop job -Dmapred.job.tracker=bigdata:9001 -kill job_201305221047_0036
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2013-05-23 12:53:34,086 Stage-1 map = 0%, reduce = 0%
2013-05-23 12:53:45,773 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 0.53 sec
2013-05-23 12:53:46,827 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 0.53 sec
2013-05-23 12:53:47,852 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 0.53 sec
MapReduce Total cumulative CPU time: 530 msec
Ended Job = job_201305221047_0036
Ended Job = -2020886930, job is filtered out (removed at runtime).
Moving data to: hdfs://bigdata:9000/tmp/hive-biadmin/hive_2013-05-23_12-53-23_368_1124072243748279142/-ext-10000
Loading data to table default.counts
Deleted hdfs://bigdata:9000/biginsights/hive/warehouse/counts
Table default.counts stats: [num_partitions: 0, num_files: 1, num_rows: 0, total_size: 290, raw_data_size: 0]
18 Rows loaded to counts
MapReduce Jobs Launched:
Job 0: Map: 1 Cumulative CPU: 0.53 sec HDFS Read: 1000 HDFS Write: 290 SUCCESS
Total MapReduce CPU Time Spent: 530 msec
OK
Time taken: 25.197 seconds
hive>
```

__18. Produce the histogram by counting the number of occurrences grouped by the names.

```
SELECT thenames, count(theids) FROM counts group by thenames;
```



```

biadmin@bigdata:/opt/ibm/biginsights/hive/bin
File Edit View Terminal Tabs Help
Number of reduce tasks not specified. Estimated from input data size: 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 mapred.reduce.tasks=<number>
Starting Job = job_201305221047_0037, Tracking URL = http://bigdata:50030/jobdetails.jsp?jobid=job_201305221047_0037
Kill Command = /ibm/BigInsights/biginsights/IHC/libexec/./bin/hadoop job -Dmapred.job.tracker=bigdata:9001 -kill job_201305221047_0037
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2013-05-23 12:56:57,165 Stage-1 map = 0%, reduce = 0%
2013-05-23 12:57:13,135 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 3.51 sec
2013-05-23 12:57:14,154 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 3.51 sec
2013-05-23 12:57:15,167 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 3.51 sec
2013-05-23 12:57:16,176 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 3.51 sec
2013-05-23 12:57:17,187 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 3.51 sec
2013-05-23 12:57:18,205 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 3.51 sec
2013-05-23 12:57:19,219 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 3.51 sec
2013-05-23 12:57:20,227 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 3.51 sec
2013-05-23 12:57:21,277 Stage-1 map = 100%, reduce = 33%, Cumulative CPU 3.51 sec
2013-05-23 12:57:22,292 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 5.75 sec
2013-05-23 12:57:23,299 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 5.75 sec
MapReduce Total cumulative CPU time: 5 seconds 750 msec
Ended Job = job_201305221047_0037
MapReduce Jobs Launched:
Job 0: Map: 1 Reduce: 1 Cumulative CPU: 5.75 sec HDFS Read: 506 HDFS Write: 127 SUCCESS
Total MapReduce CPU Time Spent: 5 seconds 750 msec
OK
BBC 2
C10 Today 2
CNN 3
Forbes 2
Healthcare IT News 2
New York Times 2
Reuters 2
The Business Journals 1
Wall Street Journal 2
Time taken: 33.469 seconds
hive>

```

The output should look similar to the image above.

__19. Quit Hive.

```
quit;
```

1.6 Summary

Congratulations! You have completed Unit 3 Working with Jaql, Pig, and Hive. You should now be able to write simple jaql and pig scripts in both the terminal and BigInsights Studio (Eclipse) as well as query and manipulate data through Hive.

NOTES

[illegible]

NOTES

[illegible]



© Copyright IBM Corporation 2013.

The information contained in these materials is provided for informational purposes only, and is provided AS IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, these materials. Nothing contained in these materials is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software. References in these materials to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. This information is based on current IBM product plans and strategy, which are subject to change by IBM without notice. Product release dates and/or capabilities referenced in these materials may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way.

IBM, the IBM logo and ibm.com are trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.



Please Recycle