Hanjiao Qiu
CSCIE63 Big Data Analytics
Assignment 05

Assignment 05 Solution

Please work in Hue's Hive Editor.

Problem 1. Create your own tables `KINGJAMES` with columns for words and frequencies and insert into the table the result of Hadoop MapReduce GREP program which produce word counts on file `all-bible`. File is provided with this assignment. Tell us all words in table `KINGJAMES` which start with letter "w" and are 4 or more characters long and appear more than 250. There are not that many of those words so you can count them by hand. However, you want to be more automated so please change your query so that it gives you the number of such words as its output. When comparing a word with a string your use `LIKE` operator, like

`word like 'a%'` or `word like '%th%'`

Symbol `'%'` means any number of characters. You measure the length of a string using function `length()` and you change the case of a word to all lower characters using function `lower()`.

Solution:

1. Copy local directory "input" into the HDFS directory. The "input" contains file "all-bible" and "all-shakespear". Run Hadoop "grep" example to count the words on file "all-bible". Check the partial content of the output file.

```
[cloudera@quickstart hw05]$ hadoop fs -put input input

[cloudera@quickstart hw05]$ hadoop jar /usr/lib/hadoop-mapreduce/hadoop-
mapreduce-examples.jar grep input/all-bible bible_freq '\w+'
16/02/28 20:18:18 INFO client.RMProxy: Connecting to ResourceManager at
/0.0.0.0:8032
16/02/28 20:18:19 WARN mapreduce.JobResourceUploader: No job jar file set.
User classes may not be found. See Job or Job#setJar(String).
16/02/28 20:18:19 INFO input.FileInputFormat: Total input paths to process : 1
16/02/28 20:18:20 INFO mapreduce.JobSubmitter: number of splits:1
16/02/28 20:18:20 INFO mapreduce.JobSubmitter: Submitting tokens for job:
job_1456675171701_0003
.....................
       File Input Format Counters
              Bytes Read=346447
       File Output Format Counters
              Bytes Written=147408

[cloudera@quickstart hw05]$ hadoop fs -ls bible_freq
Found 2 items
```

```
-rw-r--r--   1 cloudera cloudera          0 2016-02-28 20:20
bible_freq/_SUCCESS
-rw-r--r--   1 cloudera cloudera     147408 2016-02-28 20:20 bible_freq/part-r-
00000
```

Check partial content of the output file. This can be used to verify if our table is created successfully and correctly.

```
[cloudera@quickstart hw05]$ hadoop fs -cat bible_freq/part-r-00000 | head -n 20
62394  the
38985  and
34654  of
13526  to
12846  And
12603  that
12445  in
9764   shall
9672   he
8940   unto
8854   I
8385   his
8057   a
7270   for
6974   they
6913   be
6884   is
6649   him
6647   LORD
6591   not
```

2. We can do the query either through Hue Hive Query Editor or through the hive shell. We will first query through the hive shell. The usage of Hue Hive Query Editor will be shown later.

If using the hive shell, import the library that contains various tools Hive needs first. If the table we are going to create already exists, drop it first.

```
[cloudera@quickstart hw05]$ hive

Logging initialized using configuration in file:/etc/hive/conf.dist/hive-
log4j.properties
WARNING: Hive CLI is deprecated and migration to Beeline is recommended.
hive> add jar /usr/lib/hive/lib/hive-contrib.jar;
Added [/usr/lib/hive/lib/hive-contrib.jar] to class path
Added resources: [/usr/lib/hive/lib/hive-contrib.jar]
hive> DROP TABLE IF EXISTS KINGJAMES;
OK
Time taken: 0.027 seconds
hive> CREATE TABLE KINGJAMES (freq INT, word STRING) ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t' STORED AS TEXTFILE;
OK
Time taken: 0.206 seconds
```

```
hive> SHOW TABLES;
OK
kingjames
Time taken: 0.035 seconds, Fetched: 1 row(s)
hive> DESCRIBE KINGJAMES;
OK
freq                    int
word                    string
Time taken: 0.123 seconds, Fetched: 2 row(s)
```

3. Load input data "bible_freq" from HDFS. The file "part-r-00000" will be automatically moved to "/user/hive/warehouse" in HDFS. We can verify this by checking the "warehouse" directory using "hive" account.

```
hive> LOAD DATA INPATH "/user/cloudera/bible_freq" INTO TABLE KINGJAMES;
Loading data to table default.kingjames
chgrp: changing ownership of
'hdfs://quickstart.cloudera:8020/user/hive/warehouse/kingjames/part-r-00000':
User does not belong to supergroup
Table default.kingjames stats: [numFiles=1, totalSize=147408]
OK
Time taken: 1.078 seconds

[cloudera@quickstart ~]$ sudo -u hive hadoop fs -ls warehouse
Found 1 items
drwxrwxrwx   - cloudera supergroup          0 2016-02-28 21:12
warehouse/kingjames
[cloudera@quickstart ~]$ sudo -u hive hadoop fs -ls warehouse/kingjames
Found 1 items
-rwxrwxrwx   1 cloudera cloudera     147408 2016-02-28 21:09
warehouse/kingjames/part-r-00000
```

4. Select the top 20 words from the table. The words should be the same with the top 20 words we get directly from the file before. Then we can make sure the table is created correctly.

```
hive> SELECT * FROM KINGJAMES WHERE freq > 6000 SORT BY freq DESC LIMIT 20;
Query ID = cloudera_20160228211212_c43981a0-e867-4de1-ad80-82231decc806
Total jobs = 2
Launching Job 1 out of 2
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 mapreduce.job.reduces=<number>
Starting Job = job_1456675171701_0013, Tracking URL =
http://quickstart.cloudera:8088/proxy/application_1456675171701_0013/
.....................
MapReduce Total cumulative CPU time: 2 seconds 590 msec
Ended Job = job_1456675171701_0014
```

```
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 3.61 sec   HDFS Read: 153462
HDFS Write: 572 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 2.59 sec   HDFS Read: 5108
HDFS Write: 183 SUCCESS
Total MapReduce CPU Time Spent: 6 seconds 200 msec
OK
62394  the
38985  and
34654  of
13526  to
12846  And
12603  that
12445  in
9764   shall
9672   he
8940   unto
8854   I
8385   his
8057   a
7270   for
6974   they
6913   be
6884   is
6649   him
6647   LORD
6591   not
Time taken: 127.253 seconds, Fetched: 20 row(s)
```

The words are the same! The table is correct!!

5. Do our real query. Select all words in the table which start with letter "w" and are 4 or more characters long and appear more than 250 times. I will first select out all the words that meet the requirement. Then I will count their numbers. Just to cross check our results.

"SELECT *" will show all the columns for each row. We can see that all the selected words are converted to lower cases by using "lower()" and then start with "w" (compare using "like"). They are all 4 or more characters long (measure using "length()") and appears more than 250 times (check the "freq" column). We have 28 rows in total.

```
hive> SELECT * FROM KINGJAMES WHERE lower(word) like 'w%' AND length(word) >= 4
AND freq > 250;
OK
6057   with
4297   which
3819   will
2767   were
2487   when
1399   went
732    whom
694    word
```

```
652    what
546    words
512    work
443    would
436    without
407    wife
396    water
355    woman
349    When
343    wicked
335    What
335    where
304    wilderness
301    works
288    world
286    waters
284    whose
283    written
261    Wherefore
253    well
Time taken: 0.153 seconds, Fetched: 28 row(s)
```

"SELECT count(*)" will count all the rows that have been selected. It will only show 1 row result "28", which matches the results above!

```
hive> SELECT count(*) FROM KINGJAMES WHERE lower(word) like 'w%' AND
length(word) >= 4 AND freq > 250;
Query ID = cloudera_20160228212626_345463f5-3c4d-4924-9f26-6aea8852bed5
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_1456675171701_0020, Tracking URL =
http://quickstart.cloudera:8088/proxy/application_1456675171701_0020/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1456675171701_0020
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2016-02-28 21:26:23,494 Stage-1 map = 0%,  reduce = 0%
2016-02-28 21:26:42,028 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.45
sec
2016-02-28 21:26:59,299 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 4.05
sec
MapReduce Total cumulative CPU time: 4 seconds 50 msec
Ended Job = job_1456675171701_0020
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 4.05 sec   HDFS Read: 155727
HDFS Write: 3 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 50 msec
OK
```

```
28
Time taken: 57.618 seconds, Fetched: 1 row(s)
```

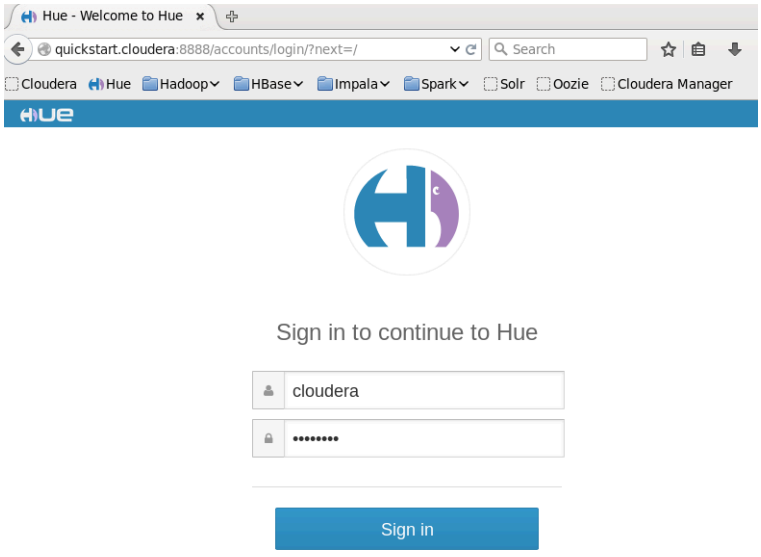
Sometimes the execution times varies a little bit. It depends on the system CPU usage.

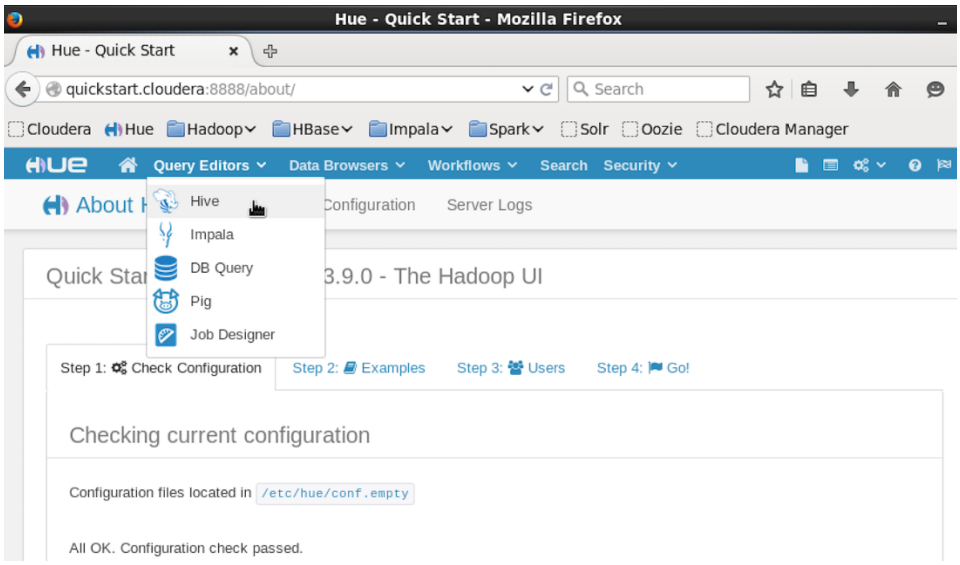
```
2016-03-03 22:06:04,048 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 3.59
sec
MapReduce Total cumulative CPU time: 3 seconds 590 msec
Ended Job = job_1456975526930_0006
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 3.59 sec   HDFS Read: 155700
HDFS Write: 3 SUCCESS
Total MapReduce CPU Time Spent: 3 seconds 590 msec
OK
28
Time taken: 30.596 seconds, Fetched: 1 row(s)
```

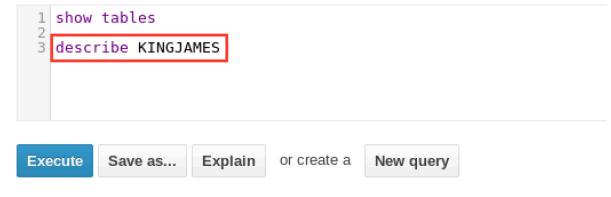
The log file below will record each steps of the above query. It has logged what is the query, when does it get start and so on. Both the Hive shell and log give out the execution time: ~30 seconds.

```
Log file: /tmp/cloudera/hive.log

Log:
2016-03-03 22:05:34,908 INFO  [main]: ql.Driver (Driver.java:execute(1316)) -
Starting command(queryId=cloudera_20160303220505_c37178ad-7f4e-460c-9aca-
e13553227087): SELECT count(*) FROM KINGJAMES WHERE lower(word) like 'w%' AND
length(word) >= 4 AND freq > 250

2016-03-03 22:06:05,169 INFO  [main]: ql.Driver
(SessionState.java:printInfo(913)) - OK
```

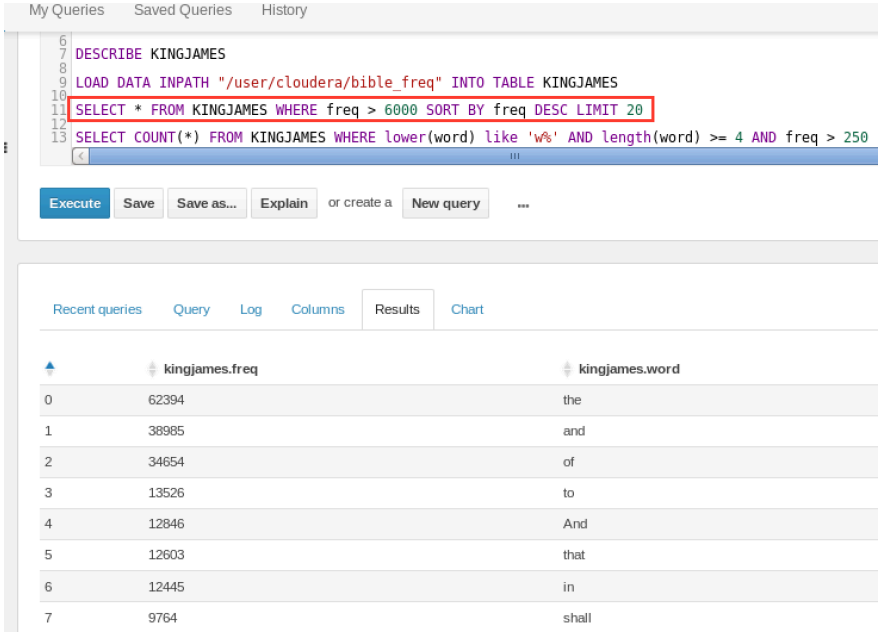
6. Run from the Hue Hive Editor. Username and password are: cloudera.



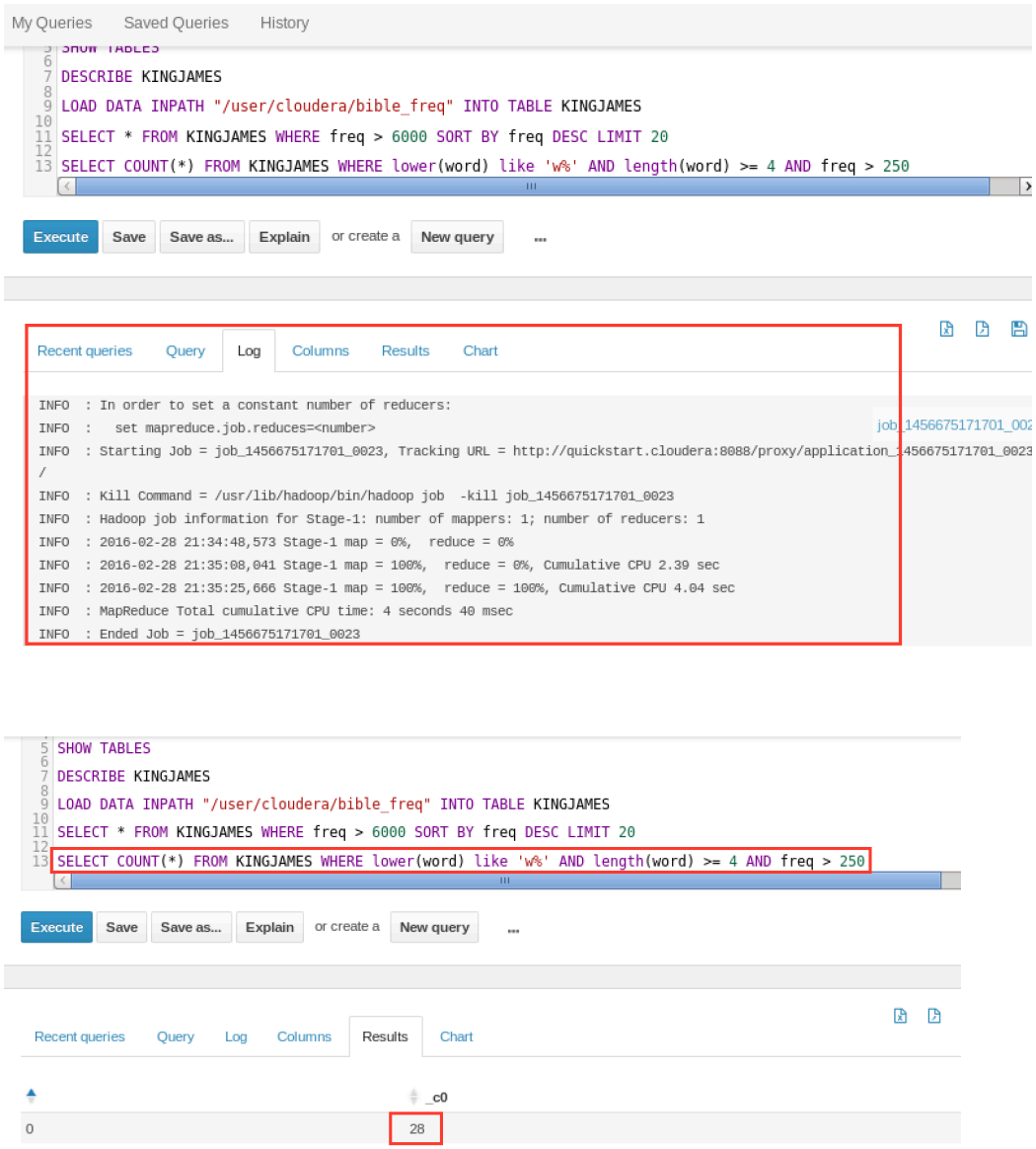




	col_name	data_type	comment
0	word	string	
1	freq	int	



	kingjames.freq	kingjames.word
0	62394	the
1	38985	and
2	34654	of
3	13526	to
4	12846	And
5	12603	that
6	12445	in
7	9764	shall



Problem 2. Create your own table `SHAKE` similar to the one we used in class and populate it with results of MapReduce GREP program applied to the file `all-shakespeare` which is provided with this assignment. Create your own `MERGED` table similar to the one we used in class. The table will list all the word and the frequencies with which they appear in either table `SHAKE` or `KINGJAMES`. Your table will be "better" than the one we used in class. In class we only inserted into that table words that appear in both texts. Please use **outer joins** to populate the table with words that also appear in one but not the other text. Tell us how many words appear in table `SHAKE` but not in `KINGJAMES` and how many appear in `KINGJAMES` and not in `SHAKE`. Select 10 words from each group for us. To solve this problem you will have to consult Hive Tutorial at https://cwiki.apache.org/confluence/display/Hive/Tutorial or simply Google around the Web.

Solution:

1. Run the "grep" example on "all-shakespeare" as Problem 1. Output the word counts to "shakespeare_freq". Check partial content of the file for validation.

```
[cloudera@quickstart hw05]$ hadoop jar /usr/lib/hadoop-mapreduce/hadoop-
mapreduce-examples.jar grep input/all-shakespeare shakespeare_freq '\w+'
16/02/28 20:04:04 INFO client.RMProxy: Connecting to ResourceManager at
/0.0.0.0:8032
16/02/28 20:04:10 WARN mapreduce.JobResourceUploader: No job jar file set.
User classes may not be found. See Job or Job#setJar(String).
16/02/28 20:04:10 INFO input.FileInputFormat: Total input paths to process : 1
16/02/28 20:04:11 INFO mapreduce.JobSubmitter: number of splits:1
16/02/28 20:04:13 INFO mapreduce.JobSubmitter: Submitting tokens for job:
job_1456675171701_0001
.....................
       File Input Format Counters
              Bytes Read=707255
       File Output Format Counters
              Bytes Written=299379

[cloudera@quickstart hw05]$ hadoop fs -ls shakespeare_freq
Found 2 items
-rw-r--r--   1 cloudera cloudera          0 2016-02-28 20:08
shakespeare_freq/_SUCCESS
-rw-r--r--   1 cloudera cloudera     299379 2016-02-28 20:08
shakespeare_freq/part-r-00000

[cloudera@quickstart hw05]$ hadoop fs -cat shakespeare_freq/part-r-00000 | head
-n 20
25578  the
23027  I
19654  and
17462  to
16444  of
13524  a
12697  you
11296  my
10699  in
8857   is
8851   that
8402   not
8033   me
8020   s
7800   And
7231   with
7165   it
6812   his
6753   be
6246   your
```

2. Create a table “SHAKE” and populate the results of “shakespeare_freq” into the table. If we don’t specify “LOCAL” when “LOAD DATA”, the data will be loaded from HDFS.

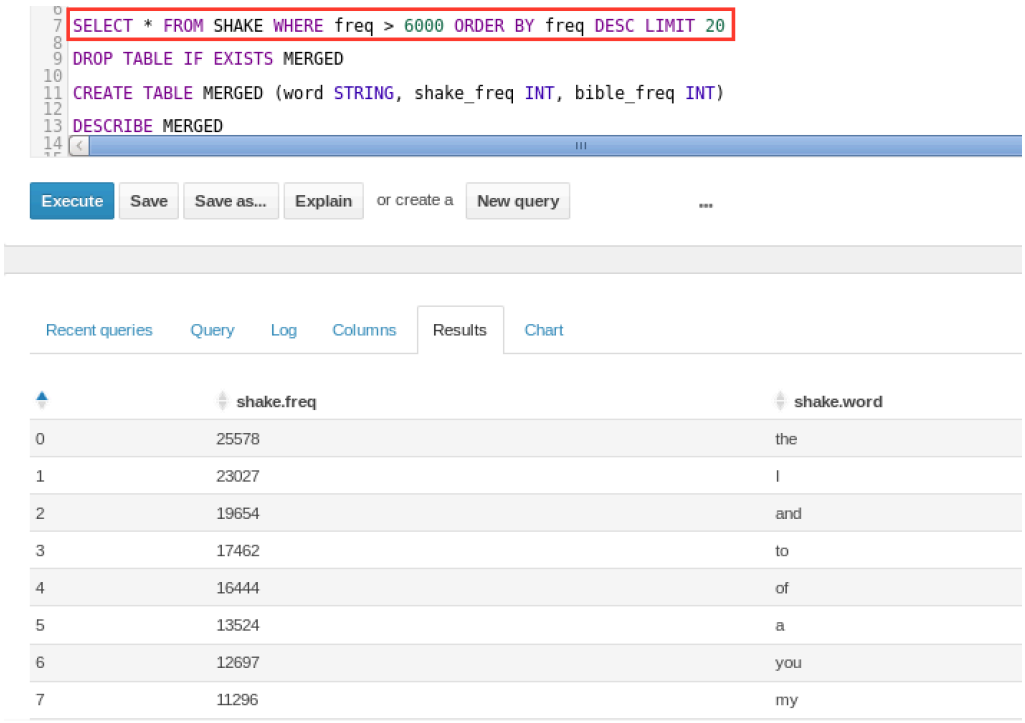
```
hive> CREATE TABLE SHAKE (freq INT, word STRING) ROW FORMAT DELIMITED FIELDS
TERMINATED BY '\t' STORED AS TEXTFILE;
OK
Time taken: 0.149 seconds
hive> SHOW TABLES;
OK
kingjames
shake
Time taken: 0.021 seconds, Fetched: 2 row(s)
hive> DESCRIBE SHAKE;
OK
freq                    int
word                    string
Time taken: 0.064 seconds, Fetched: 2 row(s)

hive> LOAD DATA INPATH "/user/cloudera/shakespeare_freq" INTO TABLE SHAKE;
Loading data to table default.shake
chgrp: changing ownership of
'hdfs://quickstart.cloudera:8020/user/hive/warehouse/shake/part-r-00000': User
does not belong to supergroup
Table default.shake stats: [numFiles=1, totalSize=299379]
OK
Time taken: 1.559 seconds
```

Select the top 20 words and check with previous results. All the words and their frequency as well as the order are the same. The table is correct!

```
hive> SELECT * FROM SHAKE WHERE freq > 6000 SORT BY freq DESC LIMIT 20;
Query ID = cloudera_20160229191111_bef6400f-c35d-4c6f-83c1-c1afc56354d6
Total jobs = 2
Launching Job 1 out of 2
.....................
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 5.37 sec   HDFS Read: 305393
HDFS Write: 565 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 2.16 sec   HDFS Read: 5093
HDFS Write: 178 SUCCESS
Total MapReduce CPU Time Spent: 7 seconds 530 msec
OK
25578  the
23027  I
19654  and
17462  to
16444  of
13524  a
12697  you
11296  my
10699  in
8857   is
```

```
8851   that
8402   not
8033   me
8020   s
7800   And
7231   with
7165   it
6812   his
6753   be
6246   your
Time taken: 79.236 seconds, Fetched: 20 row(s)
```

Do the same thing using Hue Hive Editor.



3. Create a "MERGED" table to merge "KINGJAMES" and "SHAKE". The table will list all the words and frequencies which appear in either of them.

```
hive> CREATE TABLE MERGED (word STRING, shake_freq INT, bible_freq INT);
OK
Time taken: 0.361 seconds
hive> DESCRIBE MERGED;
OK
word                    string
shake_freq              int
bible_freq              int
Time taken: 0.24 seconds, Fetched: 3 row(s)
```

The "FULL OUTER JOIN" will select the results from all rows from "KINGJAMES" and all rows from "SHAKE". "COALESCE" is a conditional function which will return

the first word that is NOT NULL in the parenthesis, or NULL if all words are NULL. Here, it will return the NULL value from (s.word, r.word). In this way, it can merge two word columns into one.

```
hive> INSERT OVERWRITE TABLE MERGED SELECT COALESCE(s.word, k.word), s.freq,
k.freq FROM SHAKE s FULL OUTER JOIN KINGJAMES k ON (s.word = k.word);
Query ID = cloudera_20160301200404_d87f5edb-c549-48ac-83ed-ecaf57346176
Total jobs = 1
Launching Job 1 out of 1
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 mapreduce.job.reduces=<number>
Starting Job = job_1456675171701_0042, Tracking URL =
http://quickstart.cloudera:8088/proxy/application_1456675171701_0042/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1456675171701_0042
Hadoop job information for Stage-1: number of mappers: 2; number of reducers: 1
2016-03-01 20:04:22,363 Stage-1 map = 0%,  reduce = 0%
2016-03-01 20:04:40,934 Stage-1 map = 33%,  reduce = 0%, Cumulative CPU 2.13
sec
2016-03-01 20:04:43,360 Stage-1 map = 50%,  reduce = 0%, Cumulative CPU 4.53
sec
2016-03-01 20:04:46,715 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 6.67
sec
2016-03-01 20:04:56,654 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU
10.71 sec
MapReduce Total cumulative CPU time: 10 seconds 710 msec
Ended Job = job_1456675171701_0042
Loading data to table default.merged
Table default.merged stats: [numFiles=1, numRows=35758, totalSize=472165,
rawDataSize=436407]
MapReduce Jobs Launched:
Stage-Stage-1: Map: 2  Reduce: 1   Cumulative CPU: 10.71 sec   HDFS Read:
458804 HDFS Write: 472242 SUCCESS
Total MapReduce CPU Time Spent: 10 seconds 710 msec
OK
Time taken: 45.366 seconds
```

We can directly select words that only belong to one table but not the other from "MERGED". We can also use "LEFT OUTER JOIN" and "RIGHT OUTER JOIN" to achieve this. I will show this approach later.

Count the words only appear in table "SHAKE" but not in "KINGJAMES". Show 10 words from them. The results are automatically sorted by "word".

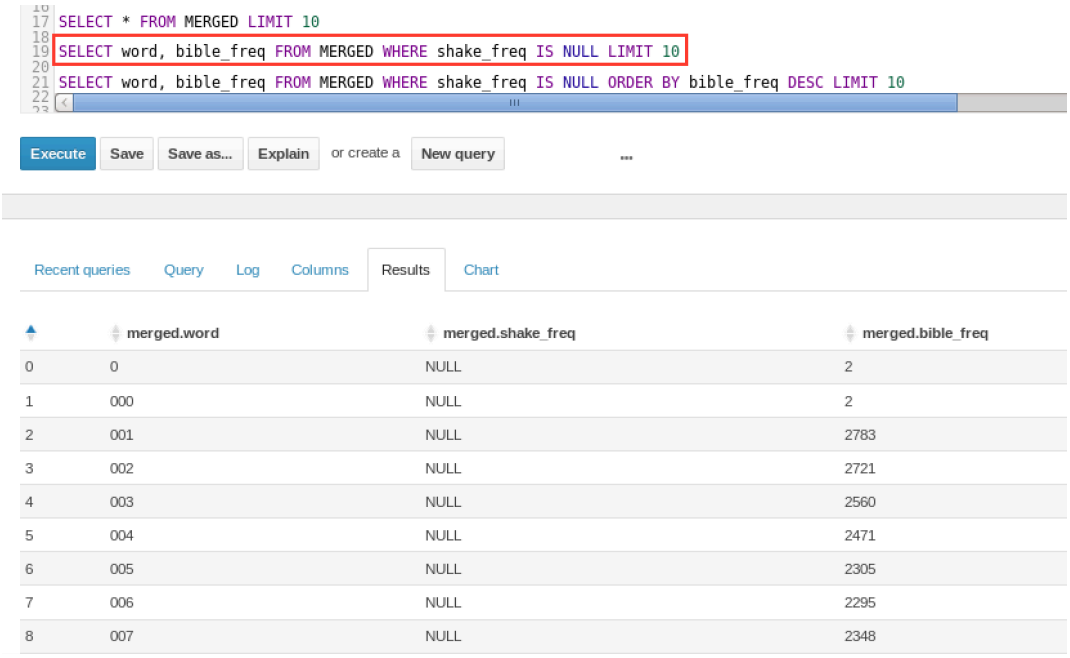
```
hive> SELECT * FROM MERGED WHERE bible_freq IS NULL LIMIT 10;
OK
2d      1       NULL
2s      2       NULL
```

```
4d      1       NULL
5s      1       NULL
6d      1       NULL
8d      1       NULL
AARON   72      NULL
ABERGAVENNY     9       NULL
ABHORSON        18      NULL
ABOUT   18      NULL
Time taken: 0.063 seconds, Fetched: 10 row(s)

hive> SELECT count(*) FROM MERGED WHERE bible_freq IS NULL;
Query ID = cloudera_20160304075959_dbd2f4da-bc9e-4399-9ee4-9203dc5f86c5
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
.....................
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 3.45 sec   HDFS Read: 479583
HDFS Write: 6 SUCCESS
Total MapReduce CPU Time Spent: 3 seconds 450 msec
OK
21428
Time taken: 28.449 seconds, Fetched: 1 row(s)
```

There are 21428 words only appear in “SHAKE”, but not in “KINGJAMES”.

Count the words only appear in table “KINGJAMES” but not in “SHAKE”. Show 10 words from them. The results are automatically sorted by “word”.

```
hive> SELECT * FROM MERGED WHERE shake_freq IS NULL LIMIT 10;
OK
0       NULL    2
000     NULL    2
001     NULL    2783
002     NULL    2721
003     NULL    2560
004     NULL    2471
005     NULL    2305
006     NULL    2295
007     NULL    2348
008     NULL    2189
Time taken: 0.08 seconds, Fetched: 10 row(s)
```



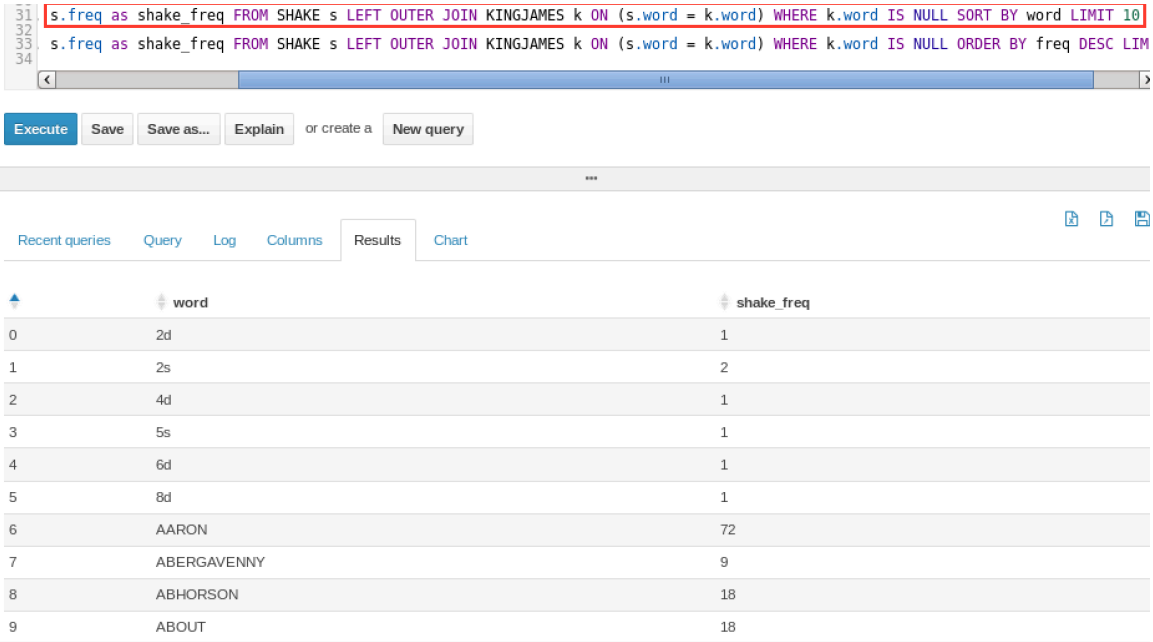

```
hive> SELECT count(*) FROM MERGED WHERE shake_freq IS NULL;
Query ID = cloudera_20160304075959_8bb8dc83-80ac-4d36-ab80-1f48f49849c7
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
.....................
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 3.37 sec   HDFS Read: 479544
HDFS Write: 5 SUCCESS
Total MapReduce CPU Time Spent: 3 seconds 370 msec
OK
6575
Time taken: 29.497 seconds, Fetched: 1 row(s)
```

There are 6575 words only appear in “KINGJAMES”, but not in “SHAKE”.

4. Do the above query using “LEFT OUTER JOIN”. It will not use the results from “MERGED”. But it should get the same results. “LEFT OUTER JOIN” will preserve the unmatched rows from the left table, joining them with a NULL row in the right table.

```
hive> SELECT s.word AS word, s.freq as freq FROM SHAKE s LEFT OUTER JOIN
KINGJAMES k ON (s.word = k.word) WHERE k.word IS NULL SORT BY word LIMIT 10;
Query ID = cloudera_20160301202323_4ff6cb14-814b-4e24-97ed-01ba23f3b4f7
Total jobs = 2
Execution log at: /tmp/cloudera/cloudera_20160301202323_4ff6cb14-814b-4e24-
97ed-01ba23f3b4f7.log
2016-03-01 08:23:59 Starting to launch local task to process map join;
      maximum memory = 1013645312
2016-03-01 08:24:00 Dump the side-table for tag: 1 with group count: 14330 into
file: file:/tmp/cloudera/eec88f65-b4e9-4eb1-b25a-df1c7573a483/hive_2016-03-
```

```
01_20-23-51_098_2624530881665304411-1/-local-10005/HashTable-Stage-2/MapJoin-
mapfile01--.hashtable
2016-03-01 08:24:00 Uploaded 1 File to: file:/tmp/cloudera/eec88f65-b4e9-4eb1-
b25a-df1c7573a483/hive_2016-03-01_20-23-51_098_2624530881665304411-1/-local-
10005/HashTable-Stage-2/MapJoin-mapfile01--.hashtable (359294 bytes)
.....................
MapReduce Jobs Launched:
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 5.7 sec   HDFS Read: 309070
HDFS Write: 327 SUCCESS
Stage-Stage-3: Map: 1  Reduce: 1   Cumulative CPU: 2.56 sec   HDFS Read: 4850
HDFS Write: 74 SUCCESS
Total MapReduce CPU Time Spent: 8 seconds 260 msec
OK
2d      1
2s      2
4d      1
5s      1
6d      1
8d      1
AARON   72
ABERGAVENNY     9
ABHORSON        18
ABOUT   18
Time taken: 72.529 seconds, Fetched: 10 row(s)
```



The above results are the same with what we've got from the "MERGED" table. Here I use "SORT BY" or "ORDER BY" to sort the results. The difference between "order by" and "sort by" is that the former guarantees total order in the output while the latter only guarantees ordering of the rows within a reducer. If there are more than one reducer, "sort by" may give partially ordered final results. Here is fine since we only have 1 reducer.

“RIGHT OUTER JOIN” will do the opposite. It will return all the rows from the right table, with the matching rows from the left table.

```
hive> SELECT k.word AS word, k.freq as freq FROM SHAKE s RIGHT OUTER JOIN
KINGJAMES k ON (s.word = k.word) WHERE s.word IS NULL SORT BY word LIMIT 10;
Query ID = cloudera_20160303224545_4ba55890-384f-4a29-995f-7e49471f27c8
Total jobs = 2
Execution log at: /tmp/cloudera/cloudera_20160303224545_4ba55890-384f-4a29-
995f-7e49471f27c8.log
2016-03-03 10:45:40 Starting to launch local task to process map join;
      maximum memory = 1013645312
2016-03-03 10:45:42 Dump the side-table for tag: 0 with group count: 29183 into
file: file:/tmp/cloudera/2b160e7a-daa7-41d1-bf5f-8c64f1272d11/hive_2016-03-
03_22-45-36_144_6288561694631611510-1/-local-10005/HashTable-Stage-2/MapJoin-
mapfile00--.hashtable
2016-03-03 10:45:42 Uploaded 1 File to: file:/tmp/cloudera/2b160e7a-daa7-41d1-
bf5f-8c64f1272d11/hive_2016-03-03_22-45-36_144_6288561694631611510-1/-local-
10005/HashTable-Stage-2/MapJoin-mapfile00--.hashtable (733215 bytes)
...................
MapReduce Jobs Launched:
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 4.39 sec   HDFS Read: 157087
HDFS Write: 330 SUCCESS
Stage-Stage-3: Map: 1  Reduce: 1   Cumulative CPU: 1.9 sec   HDFS Read: 4853
HDFS Write: 82 SUCCESS
Total MapReduce CPU Time Spent: 6 seconds 290 msec
OK
0       2
000     2
001     2783
002     2721
003     2560
004     2471
005     2305
006     2295
007     2348
008     2189
Time taken: 61.385 seconds, Fetched: 10 row(s)
```

Try sort by frequency here.

```
hive> SELECT k.word AS word, k.freq as freq FROM SHAKE s RIGHT OUTER JOIN
KINGJAMES k ON (s.word = k.word) WHERE s.word IS NULL SORT BY freq DESC LIMIT
10;
Query ID = cloudera_20160301204343_d8e82953-11a7-4911-9493-1873ad8aee6b
Total jobs = 2
Execution log at: /tmp/cloudera/cloudera_20160301204343_d8e82953-11a7-4911-
9493-1873ad8aee6b.log
2016-03-01 08:43:52 Starting to launch local task to process map join;
      maximum memory = 1013645312
2016-03-01 08:43:54 Dump the side-table for tag: 0 with group count: 29183 into
file: file:/tmp/cloudera/eec88f65-b4e9-4eb1-b25a-df1c7573a483/hive_2016-03-
01_20-43-45_620_8441776562581147209-1/-local-10005/HashTable-Stage-2/MapJoin-
mapfile50--.hashtable
```

```
2016-03-01 08:43:54 Uploaded 1 File to: file:/tmp/cloudera/eec88f65-b4e9-4eb1-
b25a-df1c7573a483/hive_2016-03-01_20-43-45_620_8441776562581147209-1/-local-
10005/HashTable-Stage-2/MapJoin-mapfile50--.hashtable (733215 bytes)
.....................
MapReduce Jobs Launched:
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 4.83 sec   HDFS Read: 157103
HDFS Write: 335 SUCCESS
Stage-Stage-3: Map: 1  Reduce: 1   Cumulative CPU: 2.41 sec   HDFS Read: 4858
HDFS Write: 89 SUCCESS
Total MapReduce CPU Time Spent: 7 seconds 240 msec
OK
001    2783
002    2721
003    2560
004    2471
19     2465
007    2348
005    2305
006    2295
009    2207
008    2189
Time taken: 74.873 seconds, Fetched: 10 row(s)

hive> SELECT count(*) FROM SHAKE s LEFT OUTER JOIN KINGJAMES k ON (s.word =
k.word) WHERE k.word IS NULL;
OK
21428
Time taken: 39.667 seconds, Fetched: 1 row(s)

hive> SELECT count(*) FROM SHAKE s RIGHT OUTER JOIN KINGJAMES k ON (s.word =
k.word) WHERE s.word IS NULL;
OK
6575
Time taken: 38.416 seconds, Fetched: 1 row(s)
```

So, there are 21428 words only appear in "SHAKE", 6575 words only appear in "KINGJAMES".

Problem 3. When you have your three queries for counting common words, words that are present in Bible but not in Shakespeare and the words present in Shakespeare but not in Bible refined and working, collect the execution times of those queries. This is not straightforward, since Hive does not give you a simple tool to time your queries. You can look in query logs (a tab next to the Results tab) and sum execution times of map and reduce jobs. That is close enough. Then change your Hue Query Editor and switch to Impala Editor. Run your queries in that editor. This time you have no way of read the time. You just make a subjective estimate. Compare the execution time of queries with Impala and Hive. Impala is usually much faster. One thing to notice here is that you can use Impala on some of Hive tables. Unfortunately not all. Hive is more versatile than Impala.

Solution:

1. Run the above query again using Hive and measure the query execution time. Here we use “JOIN” to combine rows from two tables based on the common field “word” between them. We will use “FULL OUTER JOIN” to calculated all the words appears in two books.

```
hive> SELECT count(*) FROM SHAKE s JOIN KINGJAMES k ON (s.word = k.word);
Query ID = cloudera_20160303230202_6070eb61-db5a-4bbc-9b95-365abf78d9ca
Total jobs = 1
Execution log at: /tmp/cloudera/cloudera_20160303230202_6070eb61-db5a-4bbc-
9b95-365abf78d9ca.log
2016-03-03 11:02:54 Starting to launch local task to process map join;
      maximum memory = 1013645312
2016-03-03 11:02:56 Dump the side-table for tag: 1 with group count: 14330 into
file: file:/tmp/cloudera/18df7b79-d89b-4211-bb90-50bf2e02546b/hive_2016-03-
03_23-02-49_222_5605383279257435727-1/-local-10004/HashTable-Stage-2/MapJoin-
mapfile01--.hashtable
.....................
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1456975526930_0021
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2016-03-03 23:03:06,980 Stage-2 map = 0%,  reduce = 0%
2016-03-03 23:03:15,771 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 2.02
sec
2016-03-03 23:03:24,599 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 3.29
sec
MapReduce Total cumulative CPU time: 3 seconds 290 msec
Ended Job = job_1456975526930_0021
MapReduce Jobs Launched:
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 3.29 sec   HDFS Read: 309070
HDFS Write: 5 SUCCESS
Total MapReduce CPU Time Spent: 3 seconds 290 msec
OK
7755
Time taken: 36.516 seconds, Fetched: 1 row(s)

Log file: /tmp/cloudera/hive.log

Log:
2016-03-03 23:02:50,267 INFO  [main]: ql.Driver (Driver.java:execute(1316)) -
Starting command(queryId=cloudera_20160303230202_6070eb61-db5a-4bbc-9b95-
365abf78d9ca): SELECT count(*) FROM SHAKE s JOIN KINGJAMES k ON (s.word =
k.word)

2016-03-03 23:03:25,682 INFO  [main]: ql.Driver
(SessionState.java:printInfo(913)) - Total MapReduce CPU Time Spent: 3 seconds
290 msec
2016-03-03 23:03:25,682 INFO  [main]: ql.Driver
(SessionState.java:printInfo(913)) - OK
```

The time can be found through the Hive shell as well as the hive.log.

Calculate the words appear in both books. We should use “COALESCE” here, otherwise we will duplicate the counts for words in common.

```
hive> SELECT count(COALESCE(s.word, k.word)) FROM SHAKE s FULL OUTER JOIN
KINGJAMES k ON (s.word = k.word);
Query ID = cloudera_20160303230404_ad146123-2483-4b21-9d81-c502229bdf95
Total jobs = 2
Launching Job 1 out of 2
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 mapreduce.job.reduces=<number>
Starting Job = job_1456975526930_0022, Tracking URL =
http://quickstart.cloudera:8088/proxy/application_1456975526930_0022/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1456975526930_0022
.....................

MapReduce Jobs Launched:
Stage-Stage-1: Map: 2  Reduce: 1   Cumulative CPU: 6.39 sec   HDFS Read: 457558
HDFS Write: 116 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 1.88 sec   HDFS Read: 4923
HDFS Write: 6 SUCCESS
Total MapReduce CPU Time Spent: 8 seconds 270 msec
OK
35758
Time taken: 57.397 seconds, Fetched: 1 row(s)

hive> SELECT count(*) FROM SHAKE s RIGHT OUTER JOIN KINGJAMES k ON (s.word =
k.word) WHERE s.word IS NULL;
Query ID = cloudera_20160303230505_51cfacfb-948c-48d6-837f-8bb81c50f3ed
Total jobs = 1
Execution log at: /tmp/cloudera/cloudera_20160303230505_51cfacfb-948c-48d6-
837f-8bb81c50f3ed.log
2016-03-03 11:05:19 Starting to launch local task to process map join;
      maximum memory = 1013645312
2016-03-03 11:05:20 Dump the side-table for tag: 0 with group count: 29183 into
.....................
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1456975526930_0024
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2016-03-03 23:05:30,987 Stage-2 map = 0%,  reduce = 0%
2016-03-03 23:05:39,664 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 2.3
sec
2016-03-03 23:05:47,250 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 3.49
sec
MapReduce Total cumulative CPU time: 3 seconds 490 msec
Ended Job = job_1456975526930_0024
MapReduce Jobs Launched:
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 3.49 sec   HDFS Read: 158392
HDFS Write: 5 SUCCESS
Total MapReduce CPU Time Spent: 3 seconds 490 msec
```

```
OK
6575
Time taken: 34.198 seconds, Fetched: 1 row(s)

hive> SELECT count(*) FROM SHAKE s LEFT OUTER JOIN KINGJAMES k ON (s.word =
k.word) WHERE k.word IS NULL;
Query ID = cloudera_20160303230505_3d4a13d5-8e15-43ec-a736-e1dbf87f89b7
Total jobs = 1
Execution log at: /tmp/cloudera/cloudera_20160303230505_3d4a13d5-8e15-43ec-
a736-e1dbf87f89b7.log
2016-03-03 11:05:58 Starting to launch local task to process map join;
       maximum memory = 1013645312
.....................
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1456975526930_0025
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2016-03-03 23:06:08,264 Stage-2 map = 0%,  reduce = 0%
2016-03-03 23:06:17,927 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 2.41
sec
2016-03-03 23:06:26,443 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 3.59
sec
MapReduce Total cumulative CPU time: 3 seconds 590 msec
Ended Job = job_1456975526930_0025
MapReduce Jobs Launched:
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 3.59 sec   HDFS Read: 310357
HDFS Write: 6 SUCCESS
Total MapReduce CPU Time Spent: 3 seconds 590 msec
OK
21428
Time taken: 33.363 seconds, Fetched: 1 row(s)
```

2. Do the query using Impala.

```
[cloudera@quickstart ~]$ impala-shell
Starting Impala Shell without Kerberos authentication
Connected to quickstart.cloudera:21000
Server version: impalad version 2.3.0-cdh5.5.0 RELEASE (build
0c891d79aa38f297d244855a32f1e17280e2129b)
***********************************************************************
****
Welcome to the Impala shell. Copyright (c) 2015 Cloudera, Inc. All rights
reserved.
(Impala Shell v2.3.0-cdh5.5.0 (0c891d7) built on Mon Nov  9 12:18:12 PST 2015)

The HISTORY command lists all shell commands in chronological order.
***********************************************************************
****
[quickstart.cloudera:21000] > INVALIDATE METADATA;
Query: invalidate METADATA

Fetched 0 row(s) in 2.49s
```

Required after a table is created through the Hive shell, before the table is available for Impala queries.

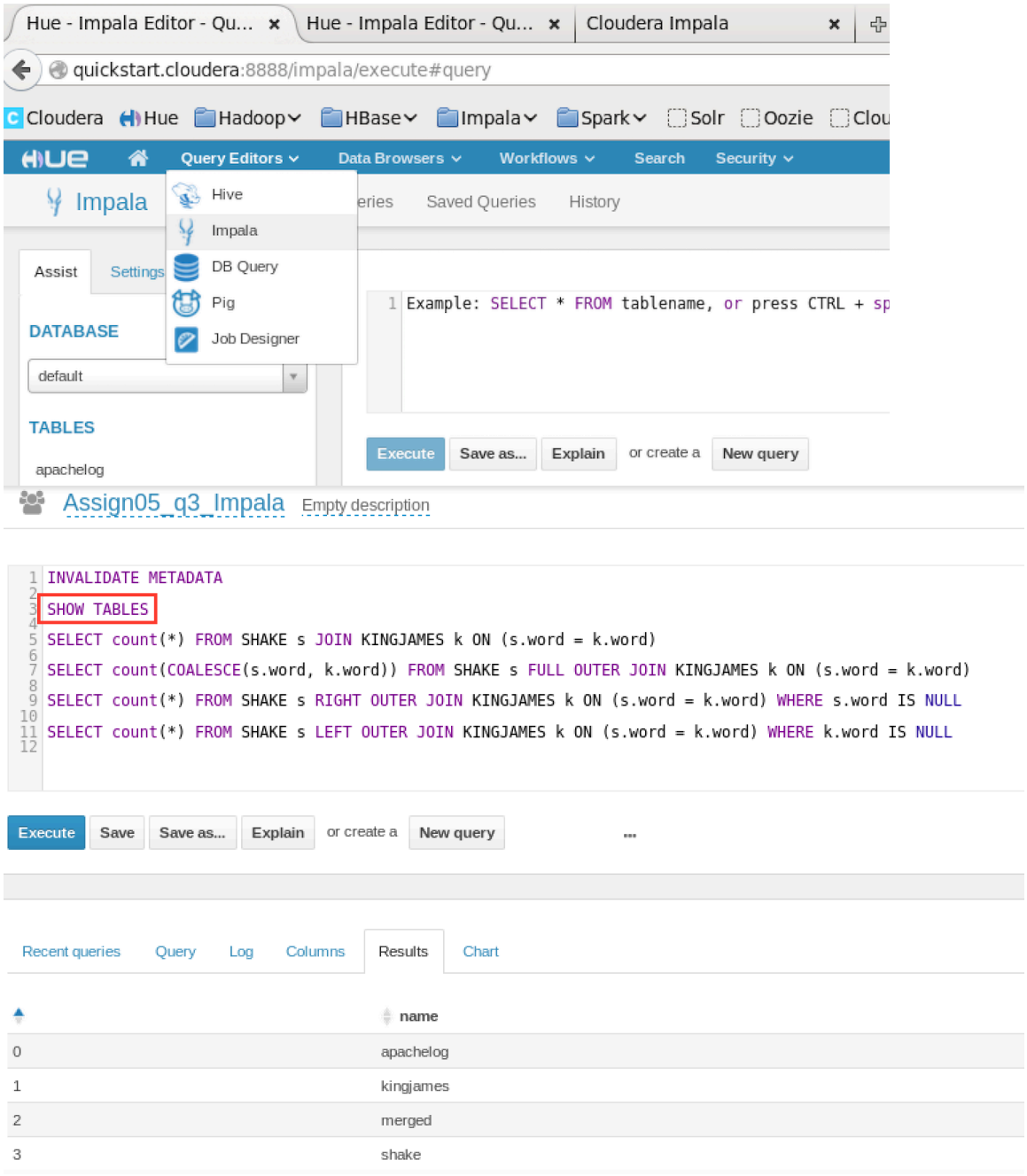
```
[quickstart.cloudera:21000] > SHOW TABLES;
Query: show TABLES
+-----------+
| name      |
+-----------+
| apachelog |
| kingjames |
| merged    |
| shake     |
+-----------+
Fetched 4 row(s) in 0.05s

[quickstart.cloudera:21000] > SELECT count(*) FROM SHAKE s JOIN KINGJAMES k ON
(s.word = k.word);
Query: select count(*) FROM SHAKE s JOIN KINGJAMES k ON (s.word = k.word)
+----------+
| count(*) |
+----------+
| 7755     |
+----------+
Fetched 1 row(s) in 3.99s
```

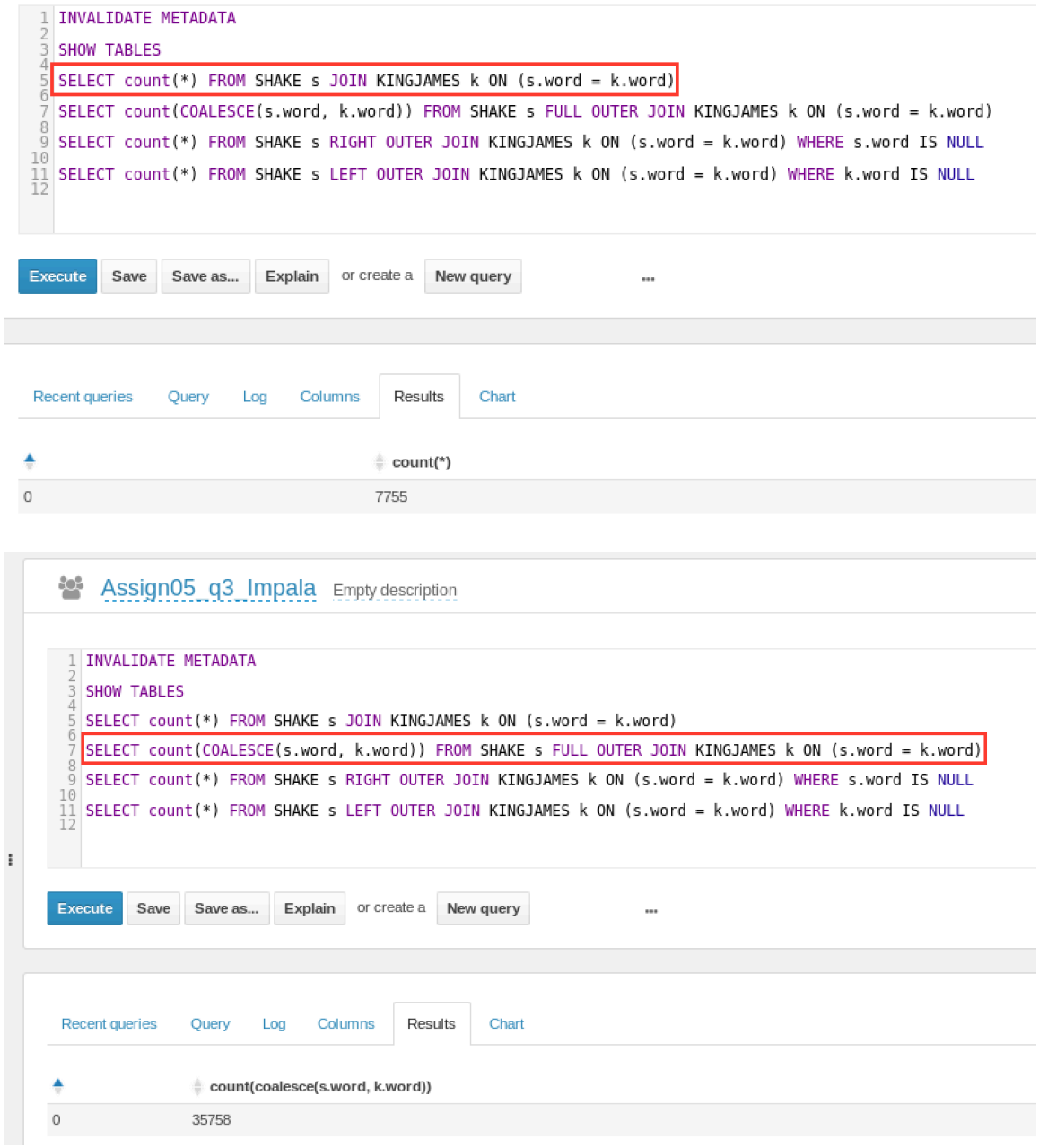
It takes slightly longer time for the first query. After loading all the data into memory, it will be faster.

```
[quickstart.cloudera:21000] > SELECT count(COALESCE(s.word, k.word)) FROM SHAKE
s FULL OUTER JOIN KINGJAMES k ON (s.word = k.word);
Query: select count(COALESCE(s.word, k.word)) FROM SHAKE s FULL OUTER JOIN
KINGJAMES k ON (s.word = k.word)
+---------------------------------+
| count(coalesce(s.word, k.word)) |
+---------------------------------+
| 35758                           |
+---------------------------------+
Fetched 1 row(s) in 1.32s

[quickstart.cloudera:21000] > SELECT count(*) FROM SHAKE s RIGHT OUTER JOIN
KINGJAMES k ON (s.word = k.word) WHERE s.word IS NULL;
Query: select count(*) FROM SHAKE s RIGHT OUTER JOIN KINGJAMES k ON (s.word =
k.word) WHERE s.word IS NULL
+----------+
| count(*) |
+----------+
| 6575     |
+----------+
Fetched 1 row(s) in 1.35s

[quickstart.cloudera:21000] > SELECT count(*) FROM SHAKE s LEFT OUTER JOIN
KINGJAMES k ON (s.word = k.word) WHERE k.word IS NULL;
Query: select count(*) FROM SHAKE s LEFT OUTER JOIN KINGJAMES k ON (s.word =
k.word) WHERE k.word IS NULL
+----------+
```

```
| count(*) |
+----------+
| 21428    |
+----------+
Fetched 1 row(s) in 1.06s
```

3. Try Impala GUI Editor.





http://localhost:25000/queries can show some logs for Impala queries. But the query execution time takes the table scanning time into account, which the total time is usually larger than what we've measured above.

Last 25 Completed Queries

User	Default Db	Statement	Query Type	Start Time	End Time	Duration	Scan Progress	State	# rows fetched	Details
cloudera	default	SELECT count(COALESCE(s.word, k.word)) FROM SHAKE s FULL OUTER JOIN KINGJAMES k ON (s.word = k.word)	QUERY	2016-03-03 23:17:26.308540000	2016-03-03 23:26:20.117499000	8m53s	2 / 2 (100%)	FINISHED	1	Details

4. Query execution time comparison between Hive and Impala.

	Hive	Impala
JOIN	36.52 s	3.99 s
FULL OUTER JOIN	57.4 s	1.32 s
LEFT OUTER JOIN	33.36 s	1.06 s
RIGHT OUTER JOIN	34.2 s	1.35 s

Impala is much faster than Hive.

Impala doesn't even use Hadoop at all. It simply has daemons running on all the nodes which cache some of the data that is in HDFS, so that these daemons can return data quickly without having to go through a whole Map/Reduce job.

There is a certain overhead involved in running a Map/Reduce job, so by short-circuiting Map/Reduce altogether we can get some pretty big gain in runtime.

Impala is faster than Apache Hive but it's not as fault tolerance and scalable as Hive. Map/Reduce materializes all intermediate results.

Problem 4. Please create Hive table APACHELOG for extraction of the content of Apache server logs:

```
CREATE TABLE apachelog (
host STRING,
identity STRING,
user STRING,
time STRING,
request STRING,
status STRING,
size STRING,
referer STRING,
agent STRING)
ROW FORMAT SERDE 'org.apache.hadoop.hive.contrib.serde2.RegexSerDe' WITH
SERDEPROPERTIES ( "input.regex" = "([^ ]*) ([^ ]*) ([^ ]*) (-|\\[[^\\]]*\\]) ([^
\"]*|\"[^\"]*\") (-|[0-9]*) (-|[0-9]*)(?: ([^ \"]*|\"[^\"]*\") ([^ \"]*|\"[^\"]*\"))?",
"output.format.string" = "%1$s %2$s %3$s %4$s %5$s %6$s %7$s %8$s %9$s" )
STORED AS TEXTFILE;
```

Please expand the above regular expression to single line before copying the entire statement to Hue Hive editor.
Test success of creation of that table using two single line samples of Apache logs contained in files `apache.access.2.log` and `apache.access.log` (note files do not have .txt suffix) contained in the attached file `examples_older.zip`. Once you are

convinced that you can safely insert those two samples into your table `apachelog`, insert a bigger log contained in file `apache_log_1.txt`. Tell us how many lines of apache logs you have in table `apachelog`.

We are also attaching two groups of example data files for Hive: `examples_older.zip` and `examples.zip`. You might find those files useful if you want to keep on learning about the technology. You could get those files by downloading Hive distributions, as described in notes.

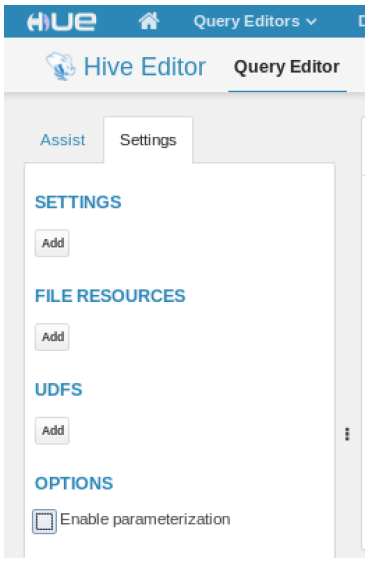
Solution:

1. Populate logs into table using Hive shell.

```
hive> CREATE TABLE apachelog (host STRING, identity STRING, user STRING, time
STRING, request STRING, status STRING, size STRING, referer STRING, agent
STRING) ROW FORMAT SERDE 'org.apache.hadoop.hive.contrib.serde2.RegexSerDe'
WITH SERDEPROPERTIES ( "input.regex" = "([^ ]*) ([^ ]*) ([^ ]*) (-
|\\[[^\\]]*\\]) ([^ \"]*|\"[^\"]*\") (-|[0-9]*) (-|[0-9]*)(?:
([^\"]*|\"[^\"]*\") ([^ \"]*|\"[^\"]*\"))?", "output.format.string" = "%1$s
%2$s %3$s %4$s %5$s %6$s %7$s %8$s %9$s" ) STORED AS TEXTFILE;
OK
Time taken: 0.118 seconds

hive> LOAD DATA LOCAL INPATH
'/mnt/hgfs/VM_shared/hw05/examples_older/apache.access.2.log' INTO TABLE
apachelog;
Loading data to table default.apachelog
Table default.apachelog stats: [numFiles=1, totalSize=219]
OK
Time taken: 0.49 seconds
hive> LOAD DATA LOCAL INPATH
'/mnt/hgfs/VM_shared/hw05/examples_older/apache.access.log' INTO TABLE
apachelog;
Loading data to table default.apachelog
Table default.apachelog stats: [numFiles=2, totalSize=305]
OK
Time taken: 0.233 seconds

hive> SELECT * FROM apachelog;
OK
127.0.0.1     -      -      [26/May/2009:00:00:00 +0000]      "GET
/someurl/?track=Blabla(Main) HTTP/1.1"  200    5864   -      "Mozilla/5.0
(Windows; U; Windows NT 6.0; en-US) AppleWebKit/525.19 (KHTML, like Gecko)
Chrome/1.0.154.65 Safari/525.19"
127.0.0.1     -      frank  [10/Oct/2000:13:55:36 -0700]      "GET /apache_pb.gif
HTTP/1.0"     202326 NULL   NULL
Time taken: 0.331 seconds, Fetched: 2 row(s)

hive> LOAD DATA LOCAL INPATH '/mnt/hgfs/VM_shared/hw05/access_log_1.txt' INTO
TABLE apachelog;
Loading data to table default.apachelog
```

```
Table default.apachelog stats: [numFiles=3, totalSize=8754422]
OK
Time taken: 0.495 seconds

hive> SELECT count(*) FROM apachelog;
Query ID = cloudera_20160301222222_9a708248-971a-4f58-a2ea-2edaea89c5e5
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_1456675171701_0065, Tracking URL =
http://quickstart.cloudera:8088/proxy/application_1456675171701_0065/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1456675171701_0065
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2016-03-01 22:22:27,387 Stage-1 map = 0%,  reduce = 0%
2016-03-01 22:22:36,333 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.82
sec
2016-03-01 22:22:45,238 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 3.16
sec
MapReduce Total cumulative CPU time: 3 seconds 160 msec
Ended Job = job_1456675171701_0065
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 3.16 sec   HDFS Read:
8762149 HDFS Write: 6 SUCCESS
Total MapReduce CPU Time Spent: 3 seconds 160 msec
OK
39346
Time taken: 30.061 seconds, Fetched: 1 row(s)


/tmp/cloudera/hive.log. 2016-03-01

2016-03-01 22:17:20,038 INFO  [main]: ql.Driver (Driver.java:execute(1316)) -
Starting command(queryId=cloudera_20160301221717_11719cb1-6d45-44ee-b61d-
da844542acd0): SELECT count(*) FROM apachelog

2016-03-01 22:22:47,376 INFO  [main]: ql.Driver
(SessionState.java:printInfo(913)) - Total MapReduce CPU Time Spent: 3 seconds
160 msec
2016-03-01 22:22:47,377 INFO  [main]: ql.Driver
(SessionState.java:printInfo(913)) - OK
```

2. Repeat the above steps using Hive Editor.




```
DROP TABLE IF EXISTS apachelog

CREATE TABLE apachelog (host STRING, identity STRING, user STRING, time STRING, request STRING, status STRING, size STRING, refere

LOAD DATA LOCAL INPATH '/mnt/hgfs/VM_shared/hw05/examples_older/apache.access.2.log' INTO TABLE apachelog

LOAD DATA LOCAL INPATH '/mnt/hgfs/VM_shared/hw05/examples_older/apache.access.log' INTO TABLE apachelog

SELECT * FROM apachelog

LOAD DATA LOCAL INPATH '/mnt/hgfs/VM_shared/hw05/access_log_1.txt' INTO TABLE apachelog

SELECT count(*) FROM apachelog
```

Execute | Save | Save as... | Explain | or create a | New query | ...

Recent queries | Query | Log | Columns | Results | Chart

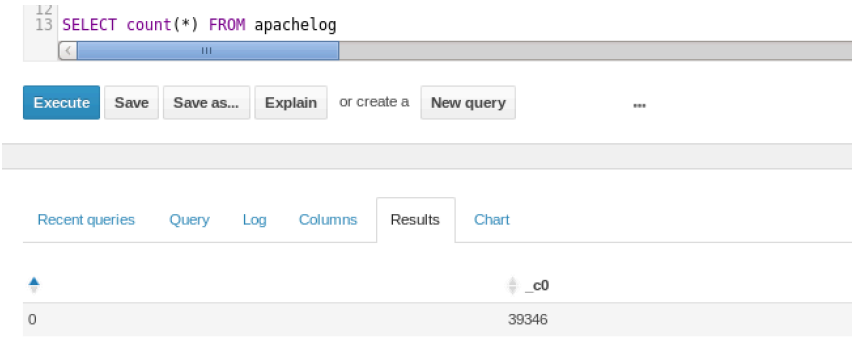
	apachelog.host	apachelog.identity	apachelog.user	apachelog.time	apachelog.request	apachelog.status	apachelog.s
0	127.0.0.1	-	-	[26/May/2009:00:00:00 +0000]	"GET /someurl/?track=Blabla(Main) HTTP/1.1"	200	5864
1	127.0.0.1	-	frank	[10/Oct/2000:13:55:36 -0700]	"GET /apache_pb.gif HTTP/1.0"	200	2326

```
SELECT count(*) FROM apachelog
```

Execute | Save | Save as... | Explain | or create a | New query | ...

Recent queries | Query | Log | Columns | Results | Chart

```
INFO  :   set hive.exec.reducers.max=<number>
INFO  : In order to set a constant number of reducers:                                                    job_1456975526930_0008
INFO  :   set mapreduce.job.reduces=<number>
INFO  : Starting Job = job_1456975526930_0008, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1456975526930_0008/
INFO  : Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1456975526930_0008
INFO  : Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
INFO  : 2016-03-03 22:18:09,788 Stage-1 map = 0%,  reduce = 0%
INFO  : 2016-03-03 22:18:18,507 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.45 sec
INFO  : 2016-03-03 22:18:27,069 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 2.54 sec
INFO  : MapReduce Total cumulative CPU time: 2 seconds 540 msec
INFO  : Ended Job = job_1456975526930_0008
```


12
13 SELECT count(*) FROM apachelog
Execute
Save
Save as...
Explain
or create a
New query
Recent queries
Query
Log
Columns
Results
Chart
_c0
0
39346