Bigdata Assignment 3

Task 1:

Write a program to implement wordcount using Pig.

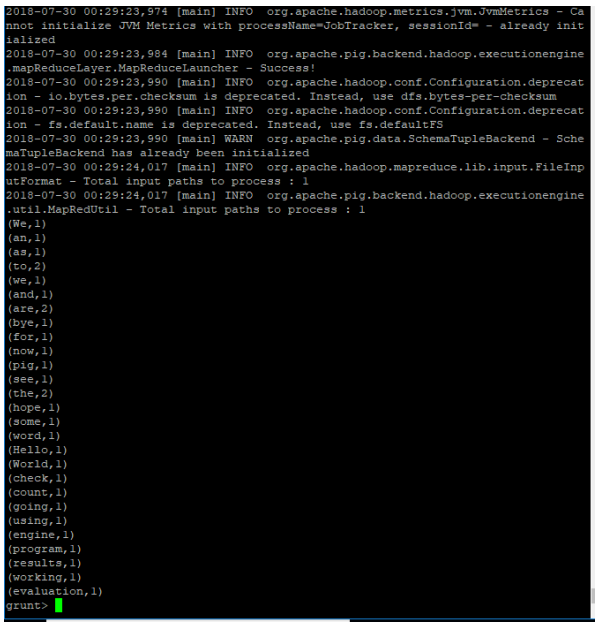
```
grunt> You have new mail in /var/spool/mail/acadgild
[acadgild@localhost ~]$ pig -x local
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/install/hadoop/hadoop-2.6.5/share
/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.clas
s]
SLF4J: Found binding in [jar:file:/home/acadgild/install/hbase/hbase-1.2.6/lib/slf
4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
18/07/30 00:16:05 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL
18/07/30 00:16:05 INFO pig.ExecTypeProvider: Picked LOCAL as the ExecType
2018-07-30 00:16:05,682 [main] INFO  org.apache.pig.Main - Apache Pig version 0.16
.0 (r1746530) compiled Jun 01 2016, 23:10:49
2018-07-30 00:16:05,683 [main] INFO  org.apache.pig.Main - Logging error messages
to: /home/acadgild/pig_1532889965679.log
2018-07-30 00:16:05,729 [main] INFO  org.apache.pig.impl.util.Utils - Default boot
up file /home/acadgild/.pigbootup not found
2018-07-30 00:16:06,250 [main] INFO  org.apache.hadoop.conf.Configuration.deprecat
ion - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address
2018-07-30 00:16:06,252 [main] INFO  org.apache.hadoop.conf.Configuration.deprecat
ion - fs.default.name is deprecated. Instead, use fs.defaultFS
2018-07-30 00:16:06,260 [main] INFO  org.apache.pig.backend.hadoop.executionengine
.HExecutionEngine - Connecting to hadoop file system at: file:///
2018-07-30 00:16:06,429 [main] INFO  org.apache.hadoop.conf.Configuration.deprecat
ion - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2018-07-30 00:16:06,493 [main] INFO  org.apache.pig.PigServer - Pig Script ID for
the session: PIG-default-9f5fa075-1303-4ea7-9423-8465a82add25
2018-07-30 00:16:06,493 [main] WARN  org.apache.pig.PigServer - ATS is disabled si
nce yarn.timeline-service.enabled set to false
grunt> clear;


grunt>text_lines = LOAD 'wc.txt' as (line:chararray);
2018-07-30 00:16:30,154 [main] INFO  org.apache.hadoop.conf.Configuration.deprecat
ion - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2018-07-30 00:16:30,155 [main] INFO  org.apache.hadoop.conf.Configuration.deprecat
ion - fs.default.name is deprecated. Instead, use fs.defaultFS
grunt>text_words = FOREACH text_lines GENERATE FLATTEN (TOKENIZE(line)) as word;
2018-07-30 00:18:09,307 [main] INFO  org.apache.pig.impl.util.SpillableMemoryManag
er - Selected heap (Tenured Gen) of size 699072512 to monitor. collectionUsageThre
shold = 489350752, usageThreshold = 489350752
grunt>text_grouped = GROUP text_words BY word;
```

```
grunt>text_lines = LOAD 'wc.txt' as (line:chararray);
2018-07-30 00:16:30,154 [main] INFO  org.apache.hadoop.conf.Configuration.deprecat
ion - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2018-07-30 00:16:30,155 [main] INFO  org.apache.hadoop.conf.Configuration.deprecat
ion - fs.default.name is deprecated. Instead, use fs.defaultFS
grunt>text_words = FOREACH text_lines GENERATE FLATTEN (TOKENIZE(line)) as word;
2018-07-30 00:18:09,307 [main] INFO  org.apache.pig.impl.util.SpillableMemoryManag
er - Selected heap (Tenured Gen) of size 699072512 to monitor. collectionUsageThre
shold = 489350752, usageThreshold = 489350752
grunt>text_grouped = GROUP text_words BY word;
grunt>word_count = FOREACH text_grouped GENERATE group, COUNT(text_words);
grunt> DUMP word_count;
2018-07-30 00:20:25,469 [main] WARN  org.apache.hadoop.util.NativeCodeLoader - Una
ble to load native-hadoop library for your platform... using builtin-java classes
where applicable
2018-07-30 00:20:25,567 [main] INFO  org.apache.pig.tools.pigstats.ScriptState - P
ig features used in the script: GROUP_BY
2018-07-30 00:20:25,692 [main] INFO  org.apache.hadoop.conf.Configuration.deprecat
ion - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2018-07-30 00:20:25,693 [main] INFO  org.apache.hadoop.conf.Configuration.deprecat
ion - fs.default.name is deprecated. Instead, use fs.defaultFS
2018-07-30 00:20:25,756 [main] INFO  org.apache.pig.newplan.logical.optimizer.Logi
calPlanOptimizer - {RULES_ENABLED=[AddForEach, ColumnMapKeyPrune, ConstantCalculat
or, GroupByConstParallelSetter, LimitOptimizer, LoadTypeCastInserter, MergeFilter,
 MergeForEach, PartitionFilterOptimizer, PredicatePushdownOptimizer, PushDownForEa
chFlatten, PushUpFilter, SplitFilter, StreamTypeCastInserter]}
2018-07-30 00:20:25,958 [main] INFO  org.apache.pig.backend.hadoop.executionengine
.mapReduceLayer.MRCompiler - File concatenation threshold: 100 optimistic? false
2018-07-30 00:20:25,989 [main] INFO  org.apache.pig.backend.hadoop.executionengine
.util.CombinerOptimizerUtil - Choosing to move algebraic foreach to combiner
2018-07-30 00:20:26,040 [main] INFO  org.apache.pig.backend.hadoop.executionengine
.mapReduceLayer.MultiQueryOptimizer - MR plan size before optimization: 1
2018-07-30 00:20:26,041 [main] INFO  org.apache.pig.backend.hadoop.executionengine
.mapReduceLayer.MultiQueryOptimizer - MR plan size after optimization: 1
2018-07-30 00:20:26,122 [main] INFO  org.apache.hadoop.conf.Configuration.deprecat
ion - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2018-07-30 00:20:26,128 [main] INFO  org.apache.hadoop.conf.Configuration.deprecat
ion - fs.default.name is deprecated. Instead, use fs.defaultFS
2018-07-30 00:20:26,162 [main] INFO  org.apache.hadoop.conf.Configuration.deprecat
ion - session.id is deprecated. Instead, use dfs.metrics.session-id
2018-07-30 00:20:26,167 [main] INFO  org.apache.hadoop.metrics.jvm.JvmMetrics - In
```

Bigdata Assignment 3

OUTPUT:



Task 2:

We have employee_details and employee_expenses files. Use local mode while running Pig and write Pig Latin script to get below results:

Step 1: Running Pig in Local mode

```
[acadgild@localhost pig_test]$ pig -x local
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/install/hadoop/hadoop-2.6.5/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/acadgild/install/hbase/hbase-1.2.6/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
18/07/17 09:12:57 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL
18/07/17 09:12:57 INFO pig.ExecTypeProvider: Picked LOCAL as the ExecType
2018-07-17 09:12:57,855 [main] INFO  org.apache.pig.Main - Apache Pig version 0.16.0 (r1746530) compiled Jun 01 2016, 23:10:49
2018-07-17 09:12:57,855 [main] INFO  org.apache.pig.Main - Logging error messages to: /home/acadgild/pig_test/pig_1531798977852.log
2018-07-17 09:12:58,034 [main] INFO  org.apache.pig.impl.util.Utils - Default bootup file /home/acadgild/.pigbootup not found
2018-07-17 09:12:58,664 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address
```

Grunt Shell opens:

```
grunt>
```

employee_details (EmpID,Name,Salary,Rating)

Bigdata Assignment 3

```
[acadgild@localhost pig_test]$ vi employee_details.txt
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost pig_test]$ cat employee_details.txt
101,Amitabh,20000,1
102,Shahrukh,10000,2
103,Akshay,11000,3
104,Anubhav,5000,4
105,Pawan,2500,5
106,Aamir,25000,1
107,Salman,17500,2
108,Ranbir,14000,3
109,Katrina,1000,4
110,Priyanka,2000,5
111,Tushar,500,1
112,Ajay,5000,2
113,Jubeen,1000,1
114,Madhuri,2000,2
```

employee_expenses(EmpID,Expense)

```
[acadgild@localhost pig_test]$ vi employee_expenses.txt
[acadgild@localhost pig_test]$ cat employee_expenses.txt
101     200
102     100
110     400
114     200
119     200
105     100
101     100
104     300
```

Step 3: Loading the “employee_details file”

```
grunt> empl = LOAD 'employee_details.txt' USING PigStorage(',') AS (emp_id:int,
emp_name:chararray, emp_salary:int,emp_rating:int);
2018-07-17 09:25:59,703 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2018-07-17 09:25:59,703 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - fs.default.name is deprecated. Instead, use fs.defaultFS
grunt> dump empl;
2018-07-17 09:26:05,495 [main] INFO  org.apache.pig.tools.pigstats.ScriptState -
 Pig features used in the script: UNKNOWN
2018-07-17 09:26:05,562 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2018-07-17 09:26:05,566 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - fs.default.name is deprecated. Instead, use fs.defaultFS
2018-07-17 09:26:05,566 [main] WARN  org.apache.pig.data.SchemaTupleBackend - Sc
hemaTupleBackend has already been initialized
2018-07-17 09:26:05,567 [main] INFO  org.apache.pig.newplan.logical.optimizer.Lo
```

Bigdata Assignment 3

```
(101,Amitabh,20000,1)
(102,Shahrukh,10000,2)
(103,Akshay,11000,3)
(104,Anubhav,5000,4)
(105,Pawan,2500,5)
(106,Aamir,25000,1)
(107,Salman,17500,2)
(108,Ranbir,14000,3)
(109,Katrina,1000,4)
(110,Priyanka,2000,5)
(111,Tushar,500,1)
(112,Ajay,5000,2)
(113,Jubeen,1000,1)
(114,Madhuri,2000,2)
grunt> describe empl;
empl: {emp_id: int,emp_name: chararray,emp_salary: int,emp_rating: int}
```

Step 4: Loading “employee_expenses.txt” file

```
grunt> emp_expl = LOAD 'employee_expenses.txt' AS (emp_id:int, expenses:int);
2018-07-17 09:29:42,742 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2018-07-17 09:29:42,742 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - fs.default.name is deprecated. Instead, use fs.defaultFS
grunt> dump emp_expl;
2018-07-17 09:29:56,572 [main] INFO  org.apache.pig.tools.pigstats.ScriptState -
 Pig features used in the script: UNKNOWN
2018-07-17 09:29:56,630 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2018-07-17 09:29:56,630 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - fs.default.name is deprecated. Instead, use fs.defaultFS
2018-07-17 09:29:56,630 [main] WARN  org.apache.pig.data.SchemaTupleBackend - Sc
hemaTupleBackend has already been initialized
2018-07-17 09:29:56,634 [main] INFO  org.apache.pig.newplan.logical.optimizer.Lo
gicalPlanOptimizer - {RULES_ENABLED=[AddForEach, ColumnMapKeyPrune, ConstantCalc
ulator, GroupByConstParallelSetter, LimitOptimizer, LoadTypeCastInserter, MergeF
```

```
(101,200)
(102,100)
(110,400)
(114,200)
(119,200)
(105,100)
(101,100)
(104,300)
grunt> describe emp_expl;
emp_expl: {emp_id: int,expenses: int}
```

a) **Top 5 employees (employee id and employee name) with highest rating. (In case two employees have same rating, employee with name coming first in dictionary should get preference)**

```
grunt> empl_with_high_rating = ORDER empl by emp_rating DESC, emp_name ASC;
grunt> empl_limit_five = LIMIT empl_with_high_rating 5;
grunt> dump empl_limit_five;
```

Bigdata Assignment 3

Output:

```
(105,Pawan,2500,5)
(110,Priyanka,2000,5)
(104,Anubhav,5000,4)
(109,Katrina,1000,4)
(103,Akshay,11000,3)
```

(b) Top 3 employees (employee id and employee name) with highest salary, whose employee id is an odd number. (In case two employees have same salary, employee with name coming first in dictionary should get preference

```
grunt> empl_salary_order = ORDER empl by emp_salary DESC;
grunt> emp_empl_id = FILTER empl by emp_id % 2 ==1;
grunt> emp_high_salary = FOREACH emp_empl_id generate emp_id,emp_name;
grunt> emp_limit_three = LIMIT emp_high_salary 3;
grunt> dump emp_limit_three;
```

Output:

```
(101,Amitabh)
(103,Akshay)
(105,Pawan)
```

(c) Employee (employee id and employee name) with maximum expense (In case two employees have same expense, employee with name coming first in dictionary should get preference)

```
grunt> empl = LOAD 'employee_details.txt' USING PigStorage(',') AS (emp_id:int,
emp_name:chararray, emp_salary:int);
2018-07-17 11:20:54,028 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2018-07-17 11:20:54,028 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - fs.default.name is deprecated. Instead, use fs.defaultFS
grunt> emp_expenses = LOAD 'employee_expenses.txt' USING PigStorage(',') AS (emp
_id:int, emp_expense:int);
2018-07-17 11:21:08,291 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2018-07-17 11:21:08,291 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - fs.default.name is deprecated. Instead, use fs.defaultFS
grunt> describe empl;
empl: {emp_id: int,emp_name: chararray,emp_salary: int}
grunt> describe emp_expenses;
emp_expenses: {emp_id: int,emp_expense: int}
```

```
grunt> join_emp_empexpense = join empl by emp_id,emp_expenses by emp_id;
grunt> max_expense = ORDER join_emp_empexpense by emp_expenses::emp_expense desc
;
grunt> Limit_maxepnse = LIMIT max_expense 1;
grunt> max_expense_final = foreach Limit_maxepnse generate empl::emp_id,empl::em
p_name;
grunt> dump max_expense_final;
```

Bigdata Assignment 3

OUTPUT:

```
(110,Priyanka)
```

(d) List of employees (employee id and employee name) having entries in employee_expenses file.

```
grunt> emp_with_exp = JOIN empl BY emp_id, emp_expenses BY emp_id;
grunt> emp_with_exp_limit = FOREACH emp_with_exp GENERATE empl::emp_id, empl::em
p_name;
grunt> emp_with_exp_distinct_data = DISTINCT emp_with_exp_limit;
grunt> dump emp_with_exp_distinct_data
```

OUTPUT:

```
(101,Amitabh)
(102,Shahrukh)
(104,Anubhav)
(105,Pawan)
(110,Priyanka)
(114,Madhuri)
```

(e) List of employees (employee id and employee name) having no entry in employee_expenses file.

```
grunt> emp_without_exp = JOIN empl BY emp_id LEFT OUTER, emp_expenses BY emp_id;
grunt> emp_without_exp_filter = FILTER emp_without_exp BY emp_expenses::emp_id i
s null;
grunt> emp_without_exp_filter_data = FOREACH emp_without_exp_filter GENERATE emp
l::emp_id, empl::emp_name;
grunt> dump emp_without_exp_filter_data;
```

Output:

```
(103,Akshay)
(106,Aamir)
(107,Salman)
(108,Ranbir)
(109,Katrina)
(111,Tushar)
(112,Ajay)
(113,Jubeen)
```

Bigdata Assignment 3

HIVE Assignment

Task 1:

Create a database named 'custom'. Create a table named temperature_data inside custom having below fields:

1. date (mm-dd-yyyy) format 2. zip code 3. temperature The table will be loaded from comma-delimited file.

Load the dataset.txt (which is ',' delimited) in the table.-

Step 1 : Ensuring database customer exists. “custom database” doesnot exists

```
hive> show databases;
OK
default
```

Step 2: Creating the database “custom”

```
hive> CREATE DATABASE custom;
OK
Time taken: 0.26 seconds
hive> show databases;
OK
custom
default
```

Step 3: Creating table “temperature_data’ with fields date, zipcode and temperature

```
hive> create table temperature_data
    > (
    > t_date string,
    > zip_code int,
    > temperature int
    > )
    > row format delimited
    > fields terminated by ',';
OK
Time taken: 2.112 seconds
```

```
hive> describe temperature_data;
OK
t_date                  string
zip_code                int
temperature             int
Time taken: 0.512 seconds, Fetched: 3 row(s)
```

Bigdata Assignment 3

Step 4: Loading data into table “temperature_data”

```
hive> load data local inpath 'temp-dataset.txt' into table temperature_data;
Loading data to table default.temperature_data
OK
Time taken: 2.666 seconds
```

```
hive> select * from temperature_data;
OK
10-01-1990      123112  10
14-02-1991      283901  11
10-03-1990      381920  15
10-01-1991      302918  22
12-02-1990      384902  9
10-01-1991      123112  11
14-02-1990      283901  12
10-03-1991      381920  16
10-01-1990      302918  23
12-02-1991      384902  10
10-01-1993      123112  11
14-02-1994      283901  12
10-03-1993      381920  16
10-01-1994      302918  23
12-02-1991      384902  10
10-01-1991      123112  11
14-02-1990      283901  12
10-03-1991      381920  16
10-01-1990      302918  23
12-02-1991      384902  10
Time taken: 0.625 seconds, Fetched: 20 row(s)
```

Task 2:

1. Fetch date and temperature from temperature_data where zip code is greater than 300000 and less than 399999.

```
hive> select t_date, temperature from temperature_data
    > where zip_code > 300000 and zip_code < 399999
    > ;
```

OUTPUT:

```
10-03-1990      15
10-01-1991      22
12-02-1990      9
10-03-1991      16
10-01-1990      23
12-02-1991      10
10-03-1993      16
10-01-1994      23
12-02-1991      10
10-03-1991      16
10-01-1990      23
12-02-1991      10
Time taken: 0.871 seconds, Fetched: 12 row(s)
```

Bigdata Assignment 3

2. Calculate maximum temperature corresponding to every year from temperature_data table. Where MapReduce launches as using aggregate functions like “max”

```
hive> select substr(t_date,7) as year , max(temperature) as maxtemp
    > from temperature_data group by substr(t_date,7);
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the futu
re versions. Consider using a different execution engine (i.e. spark, tez) or us
ing Hive 1.X releases.
Query ID = acadgild_20180805130103_901a8882-ea53-48c9-b0fb-b4d22e2a6fc5
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_1533450969481_0001, Tracking URL = http://localhost:8088/prox
y/application_1533450969481_0001/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill
job_1533450969481_0001
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2018-08-05 13:02:12,127 Stage-1 map = 0%,  reduce = 0%
```

OUTPUT:

```
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2018-08-05 13:02:12,127 Stage-1 map = 0%,  reduce = 0%
2018-08-05 13:02:31,383 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 3.22 se
c
2018-08-05 13:03:03,973 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 4.7 se
c
2018-08-05 13:03:06,427 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 6.29
sec
MapReduce Total cumulative CPU time: 6 seconds 290 msec
Ended Job = job_1533450969481_0001
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 6.29 sec   HDFS Read: 9076 HD
FS Write: 167 SUCCESS
Total MapReduce CPU Time Spent: 6 seconds 290 msec
OK
1990    23
1991    22
1993    16
1994    23
Time taken: 125.249 seconds, Fetched: 4 row(s)
```

Bigdata Assignment 3

3. Calculate maximum temperature from temperature_data table corresponding to those years which have at least 2 entries in the table.

```
hive> select substr(t_date,7) as year , max(temperature) as maxtemp
    > from temperature_data group by substr(t_date,7)
    > having COUNT(substr(t_date,7))>=2;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the futu
re versions. Consider using a different execution engine (i.e. spark, tez) or us
ing Hive 1.X releases.
Query ID = acadgild_20180805130809_081aa92e-87bf-4c5e-8c55-fc9f57883266
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_1533450969481_0002, Tracking URL = http://localhost:8088/prox
y/application_1533450969481_0002/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill
job_1533450969481_0002
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
```

OUTPUT:

```
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2018-08-05 13:08:31,755 Stage-1 map = 0%,  reduce = 0%
2018-08-05 13:08:52,277 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.99 se
c
2018-08-05 13:09:08,862 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 6.68
sec
MapReduce Total cumulative CPU time: 6 seconds 680 msec
Ended Job = job_1533450969481_0002
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 6.68 sec   HDFS Read: 10133 H
DFS Write: 167 SUCCESS
Total MapReduce CPU Time Spent: 6 seconds 680 msec
OK
1990    23
1991    22
1993    16
1994    23
Time taken: 61.287 seconds, Fetched: 4 row(s)
```

4. **Create a view on the top of last query, name it temperature_data_vw.**

```
hive> create view temp_data_vw as select substr(t_date,7) as year , max(temperat
ure) as maxtemp
    > from temperature_data group by substr(t_date,7)
    > having COUNT(substr(t_date,7))>=2;
OK
Time taken: 1.717 seconds
```

5. **Export contents from temperature_data_vw to a file in local file system, such that each file is '|' delimited.**

Bigdata Assignment 3

```
hive> insert overwrite local directory '/home/acadgild/maxtemp_data'
    > row format delimited fields terminated by '|'
    > select * from temp_data_vw;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the futu
re versions. Consider using a different execution engine (i.e. spark, tez) or us
ing Hive 1.X releases.
Query ID = acadgild_20180805133017_eb4ceeab-b745-4bb7-a8c2-044fd071160c
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_1533450969481_0003, Tracking URL = http://localhost:8088/prox
y/application_1533450969481_0003/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill
job_1533450969481_0003
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2018-08-05 13:30:29,083 Stage-1 map = 0%,  reduce = 0%
2018-08-05 13:30:41,198 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.79 se
c
2018-08-05 13:30:53,479 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 5.99
sec
MapReduce Total cumulative CPU time: 5 seconds 990 msec
Ended Job = job_1533450969481_0003
Moving data to local directory /home/acadgild/maxtemp_data
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 5.99 sec   HDFS Read: 9804 HD
FS Write: 32 SUCCESS
Total MapReduce CPU Time Spent: 5 seconds 990 msec
OK
```

OUTPUT :

```
[acadgild@localhost ~]$ cd maxtemp_data
[acadgild@localhost maxtemp_data]$ cat *
1990|23
1991|22
1993|16
1994|23
```

Bigdata Assignment 3

Advanced Hive:

Task 1:

This Data set is about Olympics. You can download the data set from the below link:

https://drive.google.com/open?id=0ByJLBTmJojjzV1czX3Nha0R3bTQ

DATE SET DESCRIPTION

The data set consists of the following fields.

Athlete: This field consists of the athlete name

Age: This field consists of athlete ages

Country: This fields consists of the country names which participated in Olympics

Year: This field consists of the year

Closing Date: This field consists of the closing date of ceremony

Sport: Consists of the sports name

Gold Medals: No. of Gold medals

Silver Medals: No. of Silver medals

Bronze Medals: No. of Bronze medals

Total Medals: Consists of total no. of medals

Step 1 : creating table Olympics_data1

```
hive> create table olympics_data1
    > (
    > athlete string,
    > age int,
    > country string,
    > year double,
    > closing_date string,
    > sport string,
    > gold_medals int,
    > silver_medals int,
    > bronze_medals int,
    > total_medals int
    > )
    > row format delimited
    > fields terminated by '\t';
OK
Time taken: 0.225 seconds
```

Bigdata Assignment 3

```
hive> describe olympics_data1;
OK
athlete                 string
age                     int
country                 string
year                    double
closing_date            string
sport                   string
gold_medals             int
silver_medals           int
bronze_medals           int
total_medals            int
Time taken: 0.193 seconds, Fetched: 10 row(s)
```

Step 2: loading data into Olympics_data1 table

```
hive> load data local inpath 'olympix_data.csv' into table olympics_data1;
Loading data to table default.olympics_data1
OK
Time taken: 1.466 seconds
```

1. **Write a Hive program to find the number of medals won by each country in swimming.**

   Query : select country, count (total_medals)from Olympics_data1

   Where sport = "Swimming" group by country;

```
hive> select country,count(total_medals)
    > from olympics_data1
    > where sport="Swimming"
    > group by country;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the futu
re versions. Consider using a different execution engine (i.e. spark, tez) or us
ing Hive 1.X releases.
Query ID = acadgild_20180805140724_86da6a12-74a7-41e3-b9de-471dcb8b9268
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_1533450969481_0004, Tracking URL = http://localhost:8088/prox
y/application_1533450969481_0004/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill
job_1533450969481_0004
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2018-08-05 14:07:37,962 Stage-1 map = 0%,  reduce = 0%
2018-08-05 14:07:55,297 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 3.22 se
c
2018-08-05 14:08:07,424 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 5.73
sec
MapReduce Total cumulative CPU time: 5 seconds 730 msec
Ended Job = job_1533450969481_0004
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 5.73 sec   HDFS Read: 529056
HDFS Write: 878 SUCCESS
```

Bigdata Assignment 3

OUTPUT:

```
Total MapReduce CPU Time Spent: 5 seconds 730 msec
OK
Argentina       1
Australia       92
Austria 2
Belarus 1
Brazil  7
Canada  5
China   29
Costa Rica      1
Croatia 1
Denmark 1
France  26
Germany 27
Great Britain   9
Hungary 7
Italy   13
Japan   30
Lithuania       1
Netherlands     32
Norway  2
Poland  1
Romania 4
Russia  19
Serbia  1
Slovakia        1
Slovenia        1
South Africa    8
South Korea     2
Spain   2
Sweden  7
Trinidad and Tobago     1
Tunisia 2
Ukraine 4
United States   145
Zimbabwe        2
Time taken: 43.913 seconds, Fetched: 34 row(s)
```

2. **Write a Hive program to find the number of medals that India won year wise.**

   Query executed to get the number of medals that India won year wise :

   Select year , count (total_medals) from Olympics_data1
   Where country = “India”
   Group by country;

Bigdata Assignment 3

```
hive> select year , count(total_medals)
    > from olympics_data1
    > where country="India"
    > group by year;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the futu
re versions. Consider using a different execution engine (i.e. spark, tez) or us
ing Hive 1.X releases.
Query ID = acadgild_20180805141433_b8eb7ef5-c583-4773-8864-a7766982d2ee
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_1533450969481_0005, Tracking URL = http://localhost:8088/prox
y/application_1533450969481_0005/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill
job_1533450969481_0005
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2018-08-05 14:14:45,778 Stage-1 map = 0%,  reduce = 0%
2018-08-05 14:14:56,012 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 3.25 se
c
2018-08-05 14:15:50,269 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 5.48
sec
MapReduce Total cumulative CPU time: 5 seconds 480 msec
Ended Job = job_1533450969481_0005
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 5.48 sec   HDFS Read: 529087
HDFS Write: 171 SUCCESS
```

OUTPUT :

```
Total MapReduce CPU Time Spent: 5 seconds 480 msec
OK
2000.0  1
2004.0  1
2008.0  3
2012.0  6
Time taken: 78.548 seconds, Fetched: 4 row(s)
```

3. **Write a Hive Program to find the total number of medals each country won.**

   Query executed to get the total number of medals won by each country

   Select country , sum(total_medals)
   From Olympics_data1
   Group by country;

Bigdata Assignment 3

```
hive> select country, sum(total_medals)
    > from olympics_data1
    > group by country;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the futu
re versions. Consider using a different execution engine (i.e. spark, tez) or us
ing Hive 1.X releases.
Query ID = acadgild_20180805142332_39f45147-78b1-4e99-a58e-a91f3e09a4b6
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_1533450969481_0006, Tracking URL = http://localhost:8088/prox
y/application_1533450969481_0006/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill
job_1533450969481_0006
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2018-08-05 14:23:44,551 Stage-1 map = 0%,  reduce = 0%
2018-08-05 14:23:53,653 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.24 se
c
2018-08-05 14:24:04,501 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 4.69
sec
MapReduce Total cumulative CPU time: 4 seconds 690 msec
Ended Job = job_1533450969481_0006
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 4.69 sec   HDFS Read: 528213
HDFS Write: 2742 SUCCESS
```

OUTPUT :

Bigdata Assignment 3

```
Total MapReduce CPU Time Spent: 4 seconds 690 msec
OK
Afghanistan     2
Algeria 8
Argentina       141
Armenia 10
Australia       609
Austria 91
Azerbaijan      25
Bahamas 24
Bahrain 1
Barbados        1
Belarus 97
Belgium 18
Botswana        1
Brazil  221
Bulgaria        41
Cameroon        20
Canada  370
Chile   22
China   530
Chinese Taipei  20
Colombia        13
Costa Rica      2
Croatia 81
Cuba    188
Cyprus  1
Czech Republic  81
Denmark 89
Dominican Republic      5
Ecuador 1
Egypt   8
Eritrea 1
Estonia 18
Ethiopia        29
Finland 118
France  318
Gabon   1
Georgia 23
Germany 629
Great Britain   322
Greece  59
Grenada 1
Guatemala       1
```

Bigdata Assignment 3

```
Hong Kong       3
Hungary 145
Iceland 15
India   11
Indonesia       22
Iran    24
Ireland 9
Israel  4
Italy   331
Jamaica 80
Japan   282
Kazakhstan      42
Kenya   39
Kuwait  2
Kyrgyzstan      3
Latvia  17
Lithuania       30
Macedonia       1
Malaysia        3
Mauritius       1
Mexico  38
Moldova 5
Mongolia        10
Montenegro      14
Morocco 11
Mozambique      1
Netherlands     318
New Zealand     52
Nigeria 39
North Korea     21
Norway  192
Panama  1
Paraguay        17
Poland  80
Portugal        9
Puerto Rico     2
Qatar   3
Romania 123
Russia  768
Saudi Arabia    6
Serbia  31
Serbia and Montenegro   38
Singapore       7
Slovakia        35
```

Bigdata Assignment 3

```
Slovenia        25
South Africa    25
South Korea     308
Spain   205
Sri Lanka       1
Sudan   1
Sweden  181
Switzerland     93
Syria   1
Tajikistan      3
Thailand        18
Togo    1
Trinidad and Tobago     19
Tunisia 4
Turkey  28
Uganda  1
Ukraine 143
United Arab Emirates    1
United States   1312
Uruguay 1
Uzbekistan      19
Venezuela       4
Vietnam 2
Zimbabwe        7
Time taken: 33.458 seconds, Fetched: 110 row(s)
```

4. **Write a Hive program to find the number of gold medals each country won.**
   Query executed to get the total number of gold medals each country won
   Selectcountry,sum(gold_medals)
   From Olympics_data1 group by country;

```
hive> select country , sum(gold_medals)
    > from olympics_data1
    > group by country;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the futu
re versions. Consider using a different execution engine (i.e. spark, tez) or us
ing Hive 1.X releases.
Query ID = acadgild_20180805143234_878a6e7a-6810-4a3f-8ec8-82a9f1572b2f
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_1533450969481_0007, Tracking URL = http://localhost:8088/prox
y/application_1533450969481_0007/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill
job_1533450969481_0007
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2018-08-05 14:32:46,270 Stage-1 map = 0%,  reduce = 0%
2018-08-05 14:32:55,555 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.12 se
c
2018-08-05 14:33:07,644 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 4.29
sec
MapReduce Total cumulative CPU time: 4 seconds 290 msec
Ended Job = job_1533450969481_0007
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 4.29 sec   HDFS Read: 528211
HDFS Write: 2703 SUCCESS
```

Bigdata Assignment 3

OUTPUT :

```
Total MapReduce CPU Time Spent: 4 seconds 290 msec
OK
Afghanistan     0
Algeria 2
Argentina       49
Armenia 0
Australia       163
Austria 36
Azerbaijan      6
Bahamas 11
Bahrain 0
Barbados        0
Belarus 17
Belgium 2
Botswana        0
Brazil  46
Bulgaria        8
Cameroon        20
Canada  168
Chile   3
China   234
Chinese Taipei  2
Colombia        2
Costa Rica      0
Croatia 35
Cuba    57
Cyprus  0
Czech Republic  14
Denmark 46
Dominican Republic      3
Ecuador 0
Egypt   1
Eritrea 0
Estonia 6
Ethiopia        13
Finland 11
France  108
Gabon   0
Georgia 6
Germany 223
Great Britain   124
Greece  12
Grenada 1
Guatemala       0
```

Bigdata Assignment 3

```
Hong Kong       0
Hungary 77
Iceland 0
India   1
Indonesia       5
Iran    10
Ireland 1
Israel  1
Italy   86
Jamaica 24
Japan   57
Kazakhstan      13
Kenya   11
Kuwait  0
Kyrgyzstan      0
Latvia  3
Lithuania       5
Macedonia       0
Malaysia        0
Mauritius       0
Mexico  19
Moldova 0
Mongolia        2
Montenegro      0
Morocco 2
Mozambique      1
Netherlands     101
New Zealand     18
Nigeria 6
North Korea     6
Norway  97
Panama  1
Paraguay        0
Poland  20
Portugal        1
Puerto Rico     0
Qatar   0
Romania 57
Russia  234
Saudi Arabia    0
Serbia  1
Serbia and Montenegro   11
Singapore       0
Slovakia        10
```

Bigdata Assignment 3

```
Slovenia        5
South Africa    10
South Korea     110
Spain   19
Sri Lanka       0
Sudan   0
Sweden  57
Switzerland     21
Syria   0
Tajikistan      0
Thailand        6
Togo    0
Trinidad and Tobago     1
Tunisia 2
Turkey  9
Uganda  1
Ukraine 31
United Arab Emirates    1
United States   552
Uruguay 0
Uzbekistan      5
Venezuela       1
Vietnam 0
Zimbabwe        2
Time taken: 34.712 seconds, Fetched: 110 row(s)
hive>
```

Task 2:

Write a hive UDF that implements functionality of string concat_ws(string SEP, array<string>). This UDF will accept two arguments, one string and one array of string. It will return a single string where all the elements of the array are separated by the SEP.

Step 1: Creating a text file stud_course_array.txt in local & populating the following data to load into a table

```
[acadgild@localhost ~]$ vi student_array.txt
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost ~]$ cat student_array.txt
stud_1 Big Data,Java
stud_2 NoSQL,Java,Hadoop
stud_3 AWS
stud_4 NULL
stud_5 DotNet,Java,Hadoop,Ruby
stud_6 Python,Java,Hadoop,Ruby,DotNet
stud_7 NoSQL,CSharp,Python,Java,Hadoop,Ruby
```

Bigdata Assignment 3

Step 2: Creating table stud_course

```
hive> create table stud_course1
    > (
    > stud_id int,
    > stud_name string,
    > course array<string>
    > )
    > row format delimited
    > fields terminated by '\t'
    > collection items terminated by ','
    > lines terminated by '\n'
    > stored as textfile;
OK
Time taken: 0.212 seconds
hive> describe stud_course1;
OK
stud_id                 int
stud_name               string
course                  array<string>
Time taken: 0.162 seconds, Fetched: 3 row(s)
hive>
```

Step 3: Loading data into table “stud_course1”

```
hive> load data local inpath 'student_array.txt' into table stud_course1;
Loading data to table custom.stud_course1
OK
Time taken: 1.162 seconds
hive> select * from stud_course1;
OK
1       stud_1  ["Big Data","Java"]
2       stud_2  ["NoSQL","Java","Hadoop"]
3       stud_3  ["AWS"]
4       stud_4  ["NULL"]
5       stud_5  ["DotNet","Java","Hadoop","Ruby"]
6       stud_6  ["Python","Java","Hadoop","Ruby","DotNet"]
7       stud_7  ["NoSQL","CSharp","Python","Java","Hadoop","Ruby"]
Time taken: 0.373 seconds, Fetched: 7 row(s)
```

Step 4: Adding hive-udf.jar file and creating a function concat_ws as concat_udf

```
hive> add jar hive-udf.jar
    > ;
Added [hive-udf.jar] to class path
Added resources: [hive-udf.jar]
hive> create temporary function concat_ws as 'concat_udf';
OK
Time taken: 0.033 seconds
```

Bigdata Assignment 3

Step 5: Displaying course using HIVE UDF 'CONCAT_WS' using '|' separator

```
hive> select concat_ws ('|',course) from stud_coursel;
OK
Big Data|Java
NoSQL|Java|Hadoop
AWS
NULL
DotNet|Java|Hadoop|Ruby
Python|Java|Hadoop|Ruby|DotNet
NoSQL|CSharp|Python|Java|Hadoop|Ruby
Time taken: 1.379 seconds, Fetched: 7 row(s)
```

Task 3:

Link: https://acadgild.com/blog/transactions-in-hive/ Refer the above given link for transactions in Hive and implement the operations given in the blog using your own sample data set and send us the screenshot.

Row-level Transactions Available in Hive 0.14

Step 1: Configurations: The below properties needs to be set appropriately in ***hive shell***

```
hive> set hive.support.concurrency=true;
hive> set hive.enforce.bucketing = true;
hive> set hive.exec.dynamic.partition.mode = nonstrict;
hive> set hive.txn.manager = org.apache.hadoop.hive.ql.lockmgr.DbTxnManager;
hive> set hive.compactor.initiator.on = true;
hive> set hive.compactor.worker.threads = a positive number on at least one inst
ance of the Thrift metastore service;
```

Step 2: Creating table "college"

```
hive> CREATE TABLE college(clg_id int,clg_name string,clg_loc string) clustered
by (clg_id) into 5 buckets stored as orc TBLPROPERTIES('transactional'='true');
OK
Time taken: 1.355 seconds
```

```
hive> show tables;
OK
college
stud_course
stud_coursel
Time taken: 2.911 seconds, Fetched: 3 row(s)
```

Step 3 : Inserting Data into a Hive Table

Bigdata Assignment 3

```
hive> INSERT INTO table college values(1,'nec','nlr'),(2,'vit','vlr'),(3,'srm','
chen'),(4,'lpu','del'),(5,'stanford','uk'),(6,'JNTUA','atp'),(7,'cambridge','us'
);
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the futu
re versions. Consider using a different execution engine (i.e. spark, tez) or us
ing Hive 1.X releases.
Query ID = acadgild_20180806093432_d6c0a7ab-641a-4327-a991-b3f380ff0aae
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 5
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_1533516361020_0001, Tracking URL = http://localhost:8088/prox
y/application_1533516361020_0001/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill
job_1533516361020_0001
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 5
2018-08-06 09:36:30,059 Stage-1 map = 0%,  reduce = 0%
2018-08-06 09:37:30,335 Stage-1 map = 0%,  reduce = 0%
2018-08-06 09:38:21,694 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 3.47 se
c
2018-08-06 09:39:23,456 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 3.47 se
c
2018-08-06 09:40:23,492 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 3.47 se
c
2018-08-06 09:41:23,756 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 3.47 se
c
2018-08-06 09:41:56,447 Stage-1 map = 100%,  reduce = 40%, Cumulative CPU 9.01 s
ec
2018-08-06 09:41:58,978 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 9.64 s
ec
2018-08-06 09:42:59,584 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 11.48
sec
2018-08-06 09:44:00,419 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 11.48
sec
2018-08-06 09:45:10,044 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 11.48
sec
2018-08-06 09:46:10,661 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 11.48
MapReduce Total cumulative CPU time: 36 seconds 820 msec
Ended Job = job_1533516361020_0001
Loading data to table custom.college
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 5   Cumulative CPU: 36.82 sec   HDFS Read: 27056
HDFS Write: 3996 SUCCESS
Total MapReduce CPU Time Spent: 36 seconds 820 msec
OK
Time taken: 836.073 seconds
```

OUTPUT:

```
hive> select * from college;
OK
5       stanford        uk
6       JNTUA   atp
1       nec     nlr
7       cambridge       us
2       vit     vlr
3       srm     chen
4       lpu     del
Time taken: 3.001 seconds, Fetched: 7 row(s)
```

Bigdata Assignment 3

Step 4: Updating the Data in Hive Table

used to update a row in Hive table.