Task 1.1) Write a program to implement word count using Pig.

Ans) Below are the commands I have used to find the word count in the file word-count.txt

```
lines = LOAD '/user/acadgild/hadoop/word-count.txt'  AS  (line:chararray);

words = FOREACH lines GENERATE FLATTEN(TOKENIZE(line)) as word;

grouped = GROUP words BY word;

wordcount = FOREACH grouped GENERATE group, COUNT(words);

DUMP wordcount;
```

File Edit View Search Terminal Help

```
grunt> lines = LOAD '/user/acadgild/hadoop/word-count.txt' AS (line:chararray);
2018-08-30 09:50:07,339 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - fs.default.name is deprecated. Instead, use fs.defaultFS
grunt> words= FOREACH lines GENERATE FLATTEN(TOKENIZE(line)) as word ;
2018-08-30 09:50:21,255 [main] INFO  org.apache.pig.impl.util.SpillableMemoryMan
ager - Selected heap (Tenured Gen) of size 699072512 to monitor. collectionUsage
Threshold = 489350752, usageThreshold = 489350752
grunt> grouped = GROUP words BY word ;
grunt> wordcount = FOREACH grouped GENERATE group,COUNT(words);
grunt> dump wordcount;
2018-08-30 09:51:06,059 [main] INFO  org.apache.pig.tools.pigstats.ScriptState -
 Pig features used in the script: GROUP_BY
2018-08-30 09:51:06,146 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - fs.default.name is deprecated. Instead, use fs.defaultFS
2018-08-30 09:51:06,153 [main] INFO  org.apache.pig.data.SchemaTupleBackend - Ke
y [pig.schematuple] was not set... will not generate code.
2018-08-30 09:51:06,237 [main] INFO  org.apache.pig.newplan.logical.optimizer.Lo
gicalPlanOptimizer - {RULES_ENABLED=[AddForEach, ColumnMapKeyPrune, ConstantCalc
ulator, GroupByConstParallelSetter, LimitOptimizer, LoadTypeCastInserter, MergeF
ilter, MergeForEach, PartitionFilterOptimizer, PredicatePushdownOptimizer, PushD
ownForEachFlatten, PushUpFilter, SplitFilter, StreamTypeCastInserter]}
2018-08-30 09:51:06,408 [main] INFO  org.apache.pig.backend.hadoop.executionengi
ne.mapReduceLayer.MRCompiler - File concatenation threshold: 100 optimistic? fal
se
```

Below is the output of dump wordcount;

```
ne.util.MapRedUtil - Total input paths to process : 1
(Hi,2)
(in,1)
(up,1)
(The,1)
(day,2)
(for,2)
(is:,2)
(the,2)
(to.,1)
(you,2)
(even,1)
(give,1)
(good,2)
(love,1)
(very,2)
(want,1)
(when,1)
(with,1)
(Magic,1)
(don't,1)
(every,2)
(falls,1)
(qoute,2)
(always,1)
(heart.,1)
(though,1)
(Today's,2)
(happens,1)
(-Jmstorm,1)
(stubborn,1)
(universe,1)
(one!!!!!!,2)
(morning!!!!,2)
grunt>
```

Task 1.2)

(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).

Ans) Below are the used commands to find the top 5 employees.

I) Loaded the employees details file from HDFS to the relation employee_details

```
grunt>employee_details = LOAD 'hadoop/employee_details.txt' USING
PigStorage(',') AS(EmpID:int,Name:chararray,Salary:int,Rating:int);
```

II) Sorted employee_deatils relation based on rating and name in ascending order.

```
grunt>employee = ORDER employee_details BY Rating asc, Name Asc;
```

```
(106,Aamir,25000,1)
(101,Amitabh,20000,1)
(113,Jubeen,1000,1)
(111,Tushar,500,1)
(112,Ajay,5000,2)
(114,Madhuri,2000,2)
```

(107,Salman,17500,2)
(102,Shahrukh,10000,2)
(103,Akshay,11000,3)
(108,Ranbir,14000,3)
(104,Anubhav,5000,4)
(109,Katrina,1000,4)
(105,Pawan,2500,5)
(110,Priyanka,2000,5)

III) grunt> limit_employee = limit employee 5;
(106,Aamir,25000,1)
(101,Amitabh,20000,1)
(113,Jubeen,1000,1)
(111,Tushar,500,1)
(112,Ajay,5000,2)

IV)grunt> top_employees = FOREACH limit_employee GENERATE EmpID,Name;
(106,Aamir)
(101,Amitabh)
(113,Jubeen)
(111,Tushar)
(112,Ajay)

```
grunt> employee_details = LOAD 'hadoop/employee_details.txt' USING PigStorage(',') AS (EmpID:int,Name:chararray,Salary:int,Ra
ting:int);
2018-08-30 12:10:47,908 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instea
d, use fs.defaultFS
grunt> employee = ORDER employee_details BY Rating asc, Name Asc;
grunt> limit_employee = limit employee 5;
grunt> top_employees = FOREACH limit_employee GENERATE EmpID,Name;
grunt> dump top_employees;
2018-08-30 12:11:56,638 [main] INFO  org.apache.pig.tools.pigstats.ScriptState - Pig features used in the script: ORDER_BY,LI
MIT
```

Final Result:

```
2018-08-30 12:14:48,172 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process :
2018-08-30 12:14:48,173 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to pr
cess : 1
(106,Aamir)
(101,Amitabh)
(113,Jubeen)
(111,Tushar)
(112,Ajay)
grunt>
```

(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)

Ans) Below are the set commands used along with result.

I)grunt>highsal_employees= ORDER employee_details BY Salary desc;
(106,Aamir,25000,1)
(101,Amitabh,20000,1)
(107,Salman,17500,2)
(108,Ranbir,14000,3)
(103,Akshay,11000,3)

(102,Shahrukh,10000,2)
(112,Ajay,5000,2)
(104,Anubhav,5000,4)
(105,Pawan,2500,5)
(110,Priyanka,2000,5)
(114,Madhuri,2000,2)
(109,Katrina,1000,4)
(113,Jubeen,1000,1)
(111,Tushar,500,1)

II) grunt>emp_odd = FILTER highsal_employees by EmpID%2==1;
(101,Amitabh,20000,1)
(107,Salman,17500,2)
(103,Akshay,11000,3)
(105,Pawan,2500,5)
(113,Jubeen,1000,1)
(109,Katrina,1000,4)
(111,Tushar,500,1)

III)grunt>topthree_employees = LIMIT emp_odd 3;
(101,Amitabh,20000,1)
(107,Salman,17500,2)
(103,Akshay,11000,3)

IV)grunt> final_result = FOREACH topthree_employees GENERATE EmpID,Name;
(101,Amitabh)
(107,Salman)
(103,Akshay)

```
grunt> highsal_employees= ORDER employee_details BY Salary desc;
grunt> emp_odd = FILTER highsal_employees by EmpID%2==1;
grunt> topthree_employees = LIMIT emp_odd 3;
grunt> final_result = FOREACH topthree_employees GENERATE EmpID,Name;
grunt> dump final_result;
2018-08-30 12:42:16,860 [main] INFO  org.apache.pig.tools.pigstats.ScriptState - Pig features used in the script: ORDER_BY,FI
LTER,LIMIT
```

```
2018-08-30 12:45:03,077 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to pro
cess : 1
(101,Amitabh)
(107,Salman)
(103,Akshay)
grunt>
```

(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)

Ans) grunt>employee_expenses = LOAD 'hadoop/emlpoyee_expenses.txt' USING PigStorage('\t') AS (EmpID:int,Expense:int);
(101,200)
(102,100)
(110,400)
(114,200)

(119,200)
(105,100)
(101,100)
(104,300)
(102,400)

grunt> new_table = JOIN employee_details BY EmpID, employee_expenses BY EmpID;
(101,Amitabh,20000,1,101,100)
(101,Amitabh,20000,1,101,200)
(102,Shahrukh,10000,2,102,400)
(102,Shahrukh,10000,2,102,100)
(104,Anubhav,5000,4,104,300)
(105,Pawan,2500,5,105,100)
(110,Priyanka,2000,5,110,400)
(114,Madhuri,2000,2,114,200)

grunt>maxexpen_employees = ORDER new_table BY Expense DESC;

(110,Priyanka,2000,5,110,400)
(102,Shahrukh,10000,2,102,400)
(104,Anubhav,5000,4,104,300)
(114,Madhuri,2000,2,114,200)
(101,Amitabh,20000,1,101,200)
(105,Pawan,2500,5,105,100)
(102,Shahrukh,10000,2,102,100)
(101,Amitabh,20000,1,101,100)

grunt> final_output = FOREACH maxexpen_employees GENERATE employee_expenses::EmpID as EmpID,employee_details::Name as Name;

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

```
grunt> employee_details = LOAD 'hadoop/employee_details.txt' USING PigStorage(',') AS (EmpID:int,Name:chararray,Salary:int,Ra
ting:int);
2018-08-30 13:05:57,005 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instea
d, use fs.defaultFS
grunt> employee_expenses = LOAD 'hadoop/emlpoyee_expenses.txt' USING PigStorage('\t') AS (EmpID:int,Expense:int);
2018-08-30 13:06:27,501 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instea
d, use fs.defaultFS
grunt> new_table = JOIN employee_details BY EmpID, employee_expenses BY EmpID;
grunt> maxexpen_employees = ORDER new_table BY Expense DESC;
grunt> final_output = FOREACH maxexpen_employees GENERATE employee_expenses::EmpID as EmpID,employee_details::Name as Name;
grunt>
```

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

Ans) grunt>new_relation = JOIN employee_details BY EmpID LEFT OUTER, employee_expenses BY EmpID;
(101,Amitabh,20000,1,101,100)

(101,Amitabh,20000,1,101,200)
(102,Shahrukh,10000,2,102,400)
(102,Shahrukh,10000,2,102,100)
(103,Akshay,11000,3,,)
(104,Anubhav,5000,4,104,300)
(105,Pawan,2500,5,105,100)
(106,Aamir,25000,1,,)
(107,Salman,17500,2,,)
(108,Ranbir,14000,3,,)
(109,Katrina,1000,4,,)
(110,Priyanka,2000,5,110,400)
(111,Tushar,500,1,,)
(112,Ajay,5000,2,,)
(113,Jubeen,1000,1,,)
(114,Madhuri,2000,2,114,200)

grunt> final_result = FILTER new_relation BY employee_expenses::Expense is not null;
(101,Amitabh,20000,1,101,100)
(101,Amitabh,20000,1,101,200)
(102,Shahrukh,10000,2,102,400)
(102,Shahrukh,10000,2,102,100)
(104,Anubhav,5000,4,104,300)
(105,Pawan,2500,5,105,100)
(110,Priyanka,2000,5,110,400)
(114,Madhuri,2000,2,114,200)

```
2018-08-30 13:43:41,194 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to pro
cess : 1
(101,Amitabh,20000,1,101,100)
(101,Amitabh,20000,1,101,200)
(102,Shahrukh,10000,2,102,400)
(102,Shahrukh,10000,2,102,100)
(104,Anubhav,5000,4,104,300)
(105,Pawan,2500,5,105,100)
(110,Priyanka,2000,5,110,400)
(114,Madhuri,2000,2,114,200)
grunt>
```

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

Ans) grunt> final_result1 = FILTER new_relation BY employee_expenses::Expense is null;
(103,Akshay,11000,3,,)
(106,Aamir,25000,1,,)
(107,Salman,17500,2,,)
(108,Ranbir,14000,3,,)
(109,Katrina,1000,4,,)
(111,Tushar,500,1,,)
(112,Ajay,5000,2,,)
(113,Jubeen,1000,1,,)

```
2018-08-30 13:46:30,070 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to
cess : 1
(103,Akshay,11000,3,,)
(106,Aamir,25000,1,,)
(107,Salman,17500,2,,)
(108,Ranbir,14000,3,,)
(109,Katrina,1000,4,,)
(111,Tushar,500,1,,)
(112,Ajay,5000,2,,)
(113,Jubeen,1000,1,,)
grunt>
```

Task 1.3) Implement the use case present in below blog link and share the complete steps along with screenshot(s) from your end.

https://acadgild.com/blog/aviation-data-analysis-using-apache-pig/

Problem1) Find out the top 5 most visited destinations.

Ans) REGISTER '/home/acadgild/airline_usecase/piggybank.jar';

A = load '/home/acadgild/airline_usecase/DelayedFlights.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',','NO_MULTILINE','UNIX','SKIP_INPUT_HEADER') ;

B = foreach A generate (int)$1 as year, (int)$10 as flight_num, (chararray)$17 as origin,(chararray) $18 as dest;
C = filter B by dest is not null;
D = group C by dest;
E = foreach D generate group, COUNT(C.dest);
F = order E by $1 DESC;
Result = LIMIT F 5;
A1 = load '/home/acadgild/airline_usecase/airports.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',','NO_MULTILINE','UNIX','SKIP_INPUT_HEADER')

A2 = foreach A1 generate (chararray)$0 as dest, (chararray)$2 as city, (chararray)$4 as country;
joined_table = join Result by $0, A2 by dest;
dump joined_table;

```
2018-09-06 17:14:49,090 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to pro
cess : 1
(ATL,106898,ATL,Atlanta,USA)
(DEN,63003,DEN,Denver,USA)
(DFW,70657,DFW,Dallas-Fort Worth,USA)
(LAX,59969,LAX,Los Angeles,USA)
(ORD,108984,ORD,Chicago,USA)
grunt>
```

problem2) Which month has seen the most number of cancellations due to bad weather?

Ans)REGISTER '/home/acadgild/airline_usecase/piggybank.jar';
A = load '/home/acadgild/airline_usecase/DelayedFlights.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',','NO_MULTILINE','UNIX','SKIP_INPUT_HEADER') ;

B = foreach A generate (int)$2 as month,(int)$10 as flight_num,(int)$22 as cancelled,(chararray)$23 as cancel_code;
C = filter B by cancelled == 1 AND cancel_code =='B';
D = group C by month;
E = foreach D generate group, COUNT(C.cancelled);
F= order E by $1 DESC;
Result = limit F 1; dump Result;

```
2018-09-06 18:03:49,195 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input
cess : 1
(12,250)
```

Problem 3) Top ten origins with the highest AVG departure delay.

Ans) REGISTER '/home/acadgild/airline_usecase/piggybank.jar';

A = load '/home/acadgild/airline_usecase/DelayedFlights.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',','NO_MULTILINE','UNIX','SKIP_INPUT_HEADER') ;

B1 = foreach A generate (int)$16 as dep_delay, (chararray)$17 as origin;
C1 = filter B1 by (dep_delay is not null) AND (origin is not null);
D1 = group C1 by origin;
E1 = foreach D1 generate group, AVG(C1.dep_delay);
Result = order E1 by $1 DESC;
Top_ten = limit Result 10;

Lookup = load '/home/acadgild/airline_usecase/airports.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',','NO_MULTILINE','UNIX','SKIP_INPUT_HEADER') ;

Lookup1 = foreach Lookup generate (chararray)$0 as origin, (chararray)$2 as city, (chararray)$4 as country;
Joined = join Lookup1 by origin, Top_ten by $0;
Final = foreach Joined generate $0,$1,$2,$4;
Final_Result = ORDER Final by $3 DESC;
dump Final_Result;

```
2018-09-06 18:11:22,691 [main] INFO  org.
2018-09-06 18:11:22,691 [main] INFO  org.
cess : 1
(CMX,Hancock,USA,116.1470588235294)
(PLN,Pellston,USA,93.76190476190476)
(SPI,Springfield,USA,83.84873949579831)
(ALO,Waterloo,USA,82.2258064516129)
(MQT,NA,USA,79.55665024630542)
(ACY,Atlantic City,USA,79.3103448275862)
(MOT,Minot,USA,78.66165413533835)
(HHH,NA,USA,76.53005464480874)
(EGE,Eagle,USA,74.12891986062718)
(BGM,Binghamton,USA,73.15533980582525)
grunt>
grunt>
```

Problem4) Which route (origin & destination) has seen the maximum diversion?

Ans)REGISTER '/home/acadgild/airline_usecase/piggybank.jar';

```
A = load '/home/acadgild/airline_usecase/DelayedFlights.csv' USING
org.apache.pig.piggybank.storage.CSVExcelStorage(',','NO_MULTILINE','UNIX','SKIP_INPUT_HEADER')
;

B = FOREACH A GENERATE (chararray)$17 as origin, (chararray)$18 as dest, (int)$24 as diversion;
C = FILTER B BY (origin is not null) AND (dest is not null) AND (diversion == 1);
D = GROUP C by (origin,dest);
E = FOREACH D generate group, COUNT(C.diversion);
F = ORDER E BY $1 DESC;
Result = limit F 10;
dump Result;
```

```
2018-09-06 17:37:11,426 [
2018-09-06 17:37:11,426 [
cess : 1
(ORD,LGA),39)
(DAL,HOU),35)
(DFW,LGA),33)
(ATL,LGA),32)
(ORD,SNA),31)
(SLC,SUN),31)
(MIA,LGA),31)
(BUR,JFK),29)
(HRL,HOU),28)
(BUR,DFW),25)
grunt>
```

Show databases; command displays all the databases available

```
hive> show databases;
OK
default
Time taken: 0.154 seconds, Fetched: 1 row(s)
hive>
```

Task2.1)Create database custom ;creates database with name custom.use custom; command uses custom database

```
hive> show databases;
OK
default
Time taken: 0.154 seconds, Fetched: 1 row(s)
hive> create database custom;
OK
Time taken: 1.616 seconds
hive> show databases;
OK
custom
default
Time taken: 0.383 seconds, Fetched: 2 row(s)
hive> use custom;
OK
Time taken: 0.105 seconds
hive>
```

Create table command creates table with specified fields

```
hive> create table temperature_data
    >      (
    >      full_date string,
    >      zip_code int,
    >      temperature int
    >      )
    >      row format delimited
    >      fields terminated by ',';
OK
Time taken: 1.53 seconds
hive> show tables;
OK
temperature_data
Time taken: 0.431 seconds, Fetched: 1 row(s)
hive>
```

```
File  Edit  View  Search  Terminal  Help
[acadgild@localhost ~]$ cat temperature_dataset.txt
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[acadgild@localhost ~]$
```

```
hive> load data local inpath 'temperature_dataset.txt' into table temperature_data;
Loading data to table custom.temperature_data
OK
Time taken: 3.755 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: 6.231 seconds, Fetched: 20 row(s)
```

Data for zipcode greater than 300000 and zipcode less than 399999

```
Time taken: 6.231 seconds, Fetched: 20 row(s)
hive> select full_date,temperature from temperature_data where zip_code>300000 and zip_code<399999;
OK
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: 2.466 seconds, Fetched: 12 row(s)
hive>
```

Maximum temperature corresponding to every year

```
hive> select year,MAX(t1.temperature) as temperature from (select substring(full_date,7,4) year,temperature from temperature_
data) t1 group by year;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execu
tion engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180904062900_382f5d73-9ae8-4011-bf5b-ba2cee5f4b38
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_1535993336114_0006, Tracking URL = http://localhost:8088/proxy/application_1535993336114_0006/
Kill Command = /home/acadgild/hadoop-2.7.2/bin/hadoop job  -kill job_1535993336114_0006
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2018-09-04 06:29:46,058 Stage-1 map = 0%,  reduce = 0%
2018-09-04 06:30:17,424 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 7.72 sec
2018-09-04 06:30:46,797 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 11.7 sec
2018-09-04 06:30:49,737 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 12.91 sec
MapReduce Total cumulative CPU time: 12 seconds 910 msec
Ended Job = job_1535993336114_0006
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 12.91 sec   HDFS Read: 9348 HDFS Write: 167 SUCCESS
Total MapReduce CPU Time Spent: 12 seconds 910 msec
OK
1990    23
1991    22
1993    16
1994    23
Time taken: 112.405 seconds, Fetched: 4 row(s)
```

Maximum temperature those years who have more than 2 entries

```
Time taken: 112.405 seconds, Fetched: 4 row(s)
hive> select year,MAX(t1.temperature) as temperature from (select substring(full_date,7,4) year,temperature from temperature_
data) t1 group by year having count(t1.year)>2;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execu
tion engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180904063317_0090765b-1d26-4b63-a01e-ac50abb9acae
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_1535993336114_0007, Tracking URL = http://localhost:8088/proxy/application_1535993336114_0007/
Kill Command = /home/acadgild/hadoop-2.7.2/bin/hadoop job  -kill job_1535993336114_0007
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2018-09-04 06:33:59,034 Stage-1 map = 0%,  reduce = 0%
2018-09-04 06:34:27,110 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 6.75 sec
2018-09-04 06:34:55,098 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 10.36 sec
2018-09-04 06:35:01,774 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 14.34 sec
MapReduce Total cumulative CPU time: 14 seconds 340 msec
Ended Job = job_1535993336114_0007
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 14.34 sec   HDFS Read: 10511 HDFS Write: 127 SUCCESS
Total MapReduce CPU Time Spent: 14 seconds 340 msec
OK
1990    23
1991    22
Time taken: 107.393 seconds, Fetched: 2 row(s)
```

Creating view with previous command data

```
: full_date, zip_code, temperature)
hive> create view temperature_data_vw as select year,MAX(t1.temperature) as temperature from (select substring(full_date,7,4)
 year,temperature from temperature_data) t1 group by year having count(t1.year)>2;
OK
Time taken: 0.687 seconds
```

```
hive> select * from temperature_data_vw;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execu
tion engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180904063945_5978822e-da69-4469-96e4-28fb61792ca9
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_1535993336114_0008, Tracking URL = http://localhost:8088/proxy/application_1535993336114_0008/
Kill Command = /home/acadgild/hadoop-2.7.2/bin/hadoop job  -kill job_1535993336114_0008
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2018-09-04 06:40:28,151 Stage-1 map = 0%,  reduce = 0%
2018-09-04 06:40:59,290 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 6.83 sec
2018-09-04 06:41:33,436 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 10.99 sec
2018-09-04 06:41:37,504 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 13.67 sec
MapReduce Total cumulative CPU time: 13 seconds 670 msec
Ended Job = job_1535993336114_0008
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 13.67 sec   HDFS Read: 10566 HDFS Write: 127 SUCCESS
Total MapReduce CPU Time Spent: 13 seconds 670 msec
OK
1990    23
1991    22
Time taken: 114.721 seconds, Fetched: 2 row(s)
```

Exporting data from view to local file with | delimited

```
12-02-1991,384902,10[acadgild@localhost ~]$ mkdir hivedata
[acadgild@localhost ~]$ ls
apache-flume-1.6.0-bin          jdk-8u101-linux-i586.tar.gz  pig_1536004931982.log
apache-flume-1.6.0-bin.tar.gz   jhg                          pig_1536005533611.log
apache-hive-2.1.0-bin           max-temp.txt                 pig_1536005723779.log
apache-hive-2.1.0-bin.tar.gz    max-temp.txty                pig_1536013112458.log
derby.log                       metastore_db                 pig_1536013197361.log
Desktop                         metastore_db.tmp             Public
Documents                       Music                        Softwares
Downloads                       orderedBySal                 sqoop-1.4.6.bin__hadoop-2.0.4-alpha
eclipse                         orderedBySal.pig             sqoop-1.4.6.bin__hadoop-2.0.4-alpha.tar.gz
employee_details.txt            Pictures                     temperature_dataset.txt
employee_expenses.txt           pig-0.16.0                   Templates
employee.java                   pig-0.16.0.tar.gz            test1.txt
hadoop                          pig_1470979104717.log        testappend.txt
hadoop-2.7.2                    pig_1471462105724.log        test.txt
hadoop-2.7.2.tar.gz             pig_1521175291666.log        Videos
hbase-1.0.3                     pig_1521175425511.log        wordcount.pig
hbase-1.0.3-bin.tar.gz          pig_1535996536994.log        Wordcount.pig
hivedata                        pig_1535999974342.log        workspace
jdk1.8.0_101                    pig_1536000427628.log
[acadgild@localhost ~]$ ls hivedata
[acadgild@localhost ~]$
```

```
hive> insert overwrite local directory '/home/acadgild/hivedata/output' row format delimited
    > fields terminated by '|'
    > select * from temperature_data_vw;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execu
tion engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180904064845_bfb4e000-2ffd-440a-94e3-dd18857438cf
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_1535993336114_0009, Tracking URL = http://localhost:8088/proxy/application_1535993336114_0009/
Kill Command = /home/acadgild/hadoop-2.7.2/bin/hadoop job  -kill job_1535993336114_0009
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2018-09-04 06:49:32,434 Stage-1 map = 0%,  reduce = 0%
2018-09-04 06:50:00,637 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 6.87 sec
2018-09-04 06:50:28,358 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 10.97 sec
2018-09-04 06:50:33,766 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 13.92 sec
MapReduce Total cumulative CPU time: 13 seconds 920 msec
Ended Job = job_1535993336114_0009
Moving data to local directory /home/acadgild/hivedata/output
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 13.92 sec   HDFS Read: 10271 HDFS Write: 16 SUCCESS
Total MapReduce CPU Time Spent: 13 seconds 920 msec
OK
Time taken: 111.359 seconds
```

```
[acadgild@localhost ~]$ cd hivedata
[acadgild@localhost hivedata]$ cd output
[acadgild@localhost output]$ ls
000000_0
[acadgild@localhost output]$ cat 000000_0
1990|23
1991|22
[acadgild@localhost output]$
```