Big Data- Hadoop- Session3 - Assignment

Exploring Pig:

Task 1:

Write a program to implement wordcount using Pig.

```
acadgild@localhost ~]$ hadoop fs -cat /user/acadgild/word-count.txt
8/08/03 01:36:29 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... us
ng builtin-java classes where applicable
his is training on hadoop big data by acadgild
his file is created using vi command.
his is an exaxmple to create a new file using append command.
acadgild@localhost ~]$
```

Commands

```
grunt> word_lines = LOAD 'word-count.txt' AS (lines:chararray);
2018-08-03 01:36:43,860 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defa
grunt> words = FOREACH word_lines GENERATE FLATTEN(TOKENIZE(lines)) as word;
grunt> grouped_words = GROUP words by word;
grunt> wordcount = FOREACH grouped_words GENERATE group, COUNT(words);
grunt> DUMP wordcount;
2018-08-03 01:37:28,701 [main] INFO  org.apache.pig.tools.pigstats.ScriptState - Pig features used in the script: GROUP_BY
2018-08-03 01:37:28,768 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defa
2018-08-03 01:37:28,773 [main] INFO  org.apache.pig.data.SchemaTupleBackend - Key [pig.schematuple] was not set... will not generate code.
2018-08-03 01:37:28,773 [main] INFO  org.apache.pig.newplan.logical.optimizer.LogicalPlanOptimizer - {RULES_ENABLED=[AddForEach, ColumnMapK
etter, LimitOptimizer, LoadTypeCastInserter, MergeFilter, MergeForEach, PartitionFilterOptimizer, PredicatePushdownOptimizer, PushDownForEa
stInserter]}
2018-08-03 01:37:28,782 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MRCompiler - File concatenation threshold
2018-08-03 01:37:28,783 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.CombinerOptimizerUtil - Choosing to move algebraic
2018-08-03 01:37:28,788 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - MR plan size before
2018-08-03 01:37:28,788 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - MR plan size after
2018-08-03 01:37:28,827 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defa
2018-08-03 01:37:28,829 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at localhost/127.0.0.1:8032
```

Output

```
(a,1)
(an,1)
(by,1)
(is,3)
(on,1)
(to,1)
(vi,1)
(big,1)
(new,1)
(This,3)
(data,1)
(file,2)
(using,2)
(append,1)
(create,1)
(hadoop,1)
(created,1)
(acadgild,1)
(command.,2)
(exaxmple,1)
(training,1)
grunt>
```

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:

Comment: 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/sha
re/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/s
lf4j-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 message
s to: /home/acadgild/pig_test/pig_1531798977852.log
2018-07-17 09:12:58,034 [main] INFO  org.apache.pig.impl.util.Utils - Default bo
```

```
[acadgild@localhost ~]$ vi employee_details.txt
[acadgild@localhost ~]$ 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

[acadgild@localhost ~]$ vi employee_expense.txt
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost ~]$ cat employee_expense.txt
101     200
102     100
110     400
114     200
119     200
105     100
101     100
104     300
102     400
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost ~]$
```

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> emp_with_high_rating = ORDER emp by emp_rating DESC , emp_name ASC;
grunt> emp_limit_five = LIMIT emp_with_high_rating 5;
grunt> dump emp_limit_five;
```

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> join_emp_expense = join emp2 by emp_id, emp expense by emp_id;
grunt> max expense = ORDER join emp expense by emp expenses::emp expense desc;
2018-08-03 02:22:07,471 [main] ERROR org.apache.pig.tools.grunt.Grunt - ERROR 1025:
<line 12, column 40> Invalid field projection. Projected field [emp_expenses::emp_expense] does not ex
in schema: emp2::emp_id:int,emp2::emp_name:chararray,emp2::emp_salary:int,emp_expense::emp_id:int,emp_
nse::expenses:int.
Details at logfile: /home/acadgild/pig_1533241063358.log
grunt> max expense = ORDER join emp expense by emp expense::expenses desc;
grunt> Limit maxexpense = LIMIT max expense 1;
grunt> max expense final = FOREACH Limit maxexpense generate emp2::emp_id, emp2::emp_name;
grunt> dump max expense final;
2018-08-03 02:25:15,984 [main] INFO  org.apache.pig.tools.pigstats.ScriptState - Pig features used in
script: HASH_JOIN,ORDER_BY,LIMIT
2018-08-03 02:25:16,037 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.c
sum is deprecated. Instead, use dfs.bytes-per-checksum
2018-08-03 02:25:16,037 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.nam
```

OUTPUT

```
(110,Priyanka)
```

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

```
grunt> emp_with_exp = JOIN emp2 by emp_id , expense BY emp_id;
2018-08-03 02:26:11,447 [main] ERROR org.apache.pig.tools.grunt.Grunt - ERROR 1200: Pig script failed to p
arse:
<line 15, column 37> Undefined alias: expense
Details at logfile: /home/acadgild/pig_1533241063358.log
grunt> emp_with_exp = JOIN emp2 by emp_id , emp_expense BY emp_id;
grunt> emp_with_limit_exp = FOREACH emp_with_exp GENERATE emp2::emp_id, emp2::emp_name;
grunt> emp_with_distinct_data = DISTINCT emp_with_limit_exp;
grunt> dump emp_with_distinct_data;
```

OUTPUT

```
 input paths to process : 1
(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 emp2  by emp_id LEFT OUTER, emp_expense BY emp_id;
grunt> emp_without_exp_filter = FILTER emp_witout_exp BY emp_expense::emp_id is null;
2018-08-03 02:30:48,785 [main] ERROR org.apache.pig.tools.grunt.Grunt - ERROR 1200: Pig script failed to p
arse:
<line 19, column 32> Undefined alias: emp_witout_exp
Details at logfile: /home/acadgild/pig_1533241063358.log
grunt> emp_without_exp_filter = FILTER emp_without_exp BY emp_expense::emp_id is null;
grunt> emp_without_exp_data= FOREACH emp_without_exp_filter GENERATE emp2::emp_id, emp2::emp_name;
grunt> dump emp_without_exp_data;
2018-08-03 02:32:31,453 [main] INFO  org.apache.pig.tools.pigstats.ScriptState - Pig features used in the
```

OUTPUT

```
 input paths to process : 1
(103,Akshay)
(106,Aamir)
(107,Salman)
(108,Ranbir)
(109,Katrina)
(111,Tushar)
(112,Ajay)
(113,Jubeen)
(,)
```

Task 3:

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

Problem Statement 1

Find out the top 5 most visited destinations.

Commands to execute:

```
grunt> REGISTER '/home/acadgild/airline_usecase/piggybank.jar';
2018-08-06 20:26:29,516 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
grunt> A= load'/home/acadgild/airline_usecase/DelayedFlight.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',','NO_MULTILINE','UNIX'
2018-08-06 20:26:41,312 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
grunt> B = foreach A generate (int)$1 as year, (int)$10 as flight_num, (chararray)$17 as origin, (chararray) $18 as dest;
grunt> C = filter B by dest is not null;
grunt> D = group C by dest;
grunt> E = foreach D generate group, COUNT(C.dest);
grunt> F = order E by $1 DESC;
grunt> Result = LIMIT F 5;
grunt> A1= load '/home/acadgild/airline_usecase/airport.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',','NO_MULTILINE','UNIX','SK
2018-08-06 20:27:42,641 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
grunt> A2 = foreach A1 generate (chararray)$0 as dest, (chararray) $2 as city, (chararray)$4 as country;
grunt> joined_table =join Result by $0, A2 by dest;
grunt> dump joined_table;
```

OUTPUT

```
(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)
```

Problem Statement 2

Which month has seen the greatest number of cancellations due to bad weather

Commands to execute:

```
grunt> REGISTER '/home/acadgild/airline_usecase/piggybank.jar';
grunt> A= load '/airline_usecase/DelayedFlights.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',','NO_MULTILINE','UNIX','SKIP_INPUT_HEA
DER');
2018-08-06 23:20:37,768 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
grunt> B = foreach A generate (int)$2 as month, (int)$10 as flight_num, (int)$22 as cancelled, (chararray) $23 as cancel_code;
grunt> C = filter B by cancelled == 1 AND cancel_code == 'B';
grunt> D = group C by month;
grunt> E = foreach D generate group, COUNT(C.cancelled);
grunt> F = ORDER E BY $1 DESC;
grunt> Result = limit F 1;
grunt> dump Result;
```

OUTPUT

```
2018-08-06 23:25:55,896 [main] INFO  org.apache.pig.backend.h
(12,250)
grunt>
```

Problem Statement 3

Top ten origins with the highest AVG departure delay

Commands to execute:

```
grunt> REGISTER '/home/acadgild/airline_usecase/piggybank.jar';
grunt> A = load '/airline_usecase/DelayedFlights.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',','NO_MULTILINE','UNIX','SKIP_INPUT_HE
ADER');
2018-08-06 23:00:01,337 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
grunt> B1 = fOREACH A GENERATE (int)$16 as dep_delay, (chararray)$17 as origin;
grunt> C1 = FILTER B1 BY (dep_delay is not null) AND (origin is not null);
grunt> D1 = GROUP C1 by (origin);
grunt> E1 = FOREACH D1 generate group, AVG(C1.dep_delay);
grunt> Result = order E1 by $1 DESC;
grunt> Top_ten = limit Result 10;
grunt> Lookup = load '/airline_usecase/airports.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',','NO_MULTILINE','UNIX','SKIP_INPUT_HEA
DER');
2018-08-06 23:04:05,216 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
grunt> Lookup1 = foreach Lookup generate (chararray)$0 as origin, (chararray)$2 as city, (chararray)$4 as country;
grunt> Joined = join Lookup1 by origin, Top_ten by $0;
grunt> Final = foreach Joined generate $0,$1,$2,$4;
grunt> Final_Result = ORDER Final by $3 DESC;
grunt> dump Final_Result;
```

OUTPUT

```
2018-08-06 23:11:03,174 [main] INFO  org.apache.pig.backend.h
(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>
```

Problem Statement 4

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

Commands to execute:

```
grunt> REGISTER '/home/acadgild/airline_usecase/piggybank.jar';
2018-08-06 22:49:56,657 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
grunt> A = load '/airline_usecase/DelayedFlights.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',','NO_MULTILINE','UNIX','SKIP_INPUT_HEADER');
2018-08-06 22:50:19,862 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
grunt> B = fOREACH A GENERATE (chararray)$17 as origin, (chararray)$18 as dest, (int)$24 as diversion;
grunt> C = FILTER B BY (origin is not null) AND (dest is not null) AND (diversion == 1);
grunt> D = GROUP C by (origin,dest);
grunt> E = FOREACH D generate group, COUNT(C.diversion);
grunt> F = ORDER E BY $1 DESC;
grunt> Result = limit F 10;
grunt> dump Result;
```

```
2018-08-06 22:54:44,021 [main] INFO  org.apache.hadoop.mapreduce.lib.input
2018-08-06 22:54:44,021 [main] INFO  org.apache.pig.backend.hadoop.executi
((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>
```

Hive Basics:

Task 1:

Create a database named 'custom'.

OUTPUT

```
[acadgild@localhost ~]$ sudo service mysqld start
[sudo] password for acadgild:
Starting mysqld:                                           [  OK  ]
[acadgild@localhost ~]$ hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/install/hive/apache-hive-2.3.2-bin/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/
SLF4J: Found binding in [jar:file:/home/acadgild/install/hadoop/hadoop-2.6.5/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!/
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]

Logging initialized using configuration in jar:file:/home/acadgild/install/hive/apache-hive-2.3.2-bin/lib/hive-common-2.3.2.ja
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engin
hive> create database custom
    > ;
OK
Time taken: 9.405 seconds
hive> use custom;
OK
Time taken: 0.044 seconds
hive>
```

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.-

OUTPUT

Dataset

```
[acadgild@localhost ~]$ vi temp_data.txt
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost ~]$ cat temp_data.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
[acadgild@localhost ~]$ pwd
/home/acadgild
```

Table temperature_data

```
hive> use custom;
OK
Time taken: 0.065 seconds
hive> CREATE TABLE temperature_data(
    > date_temp STRING,
    > zip_code INT,
    > temperature INT )
    > row format delimited fields terminated by ',';
OK
Time taken: 0.33 seconds
hive> Describe temperature_data;
OK
date_temp               string
zip_code                int
temperature             int
Time taken: 0.173 seconds, Fetched: 3 row(s)
hive> Load Data Local INPATH '/home/acadgild/temp_data.txt' INTO TABLE temperature_data;
Loading data to table custom.temperature_data
OK
Time taken: 1.268 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
Time taken: 0.31 seconds, Fetched: 10 row(s)
```

Task 2:

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

OUTPUT

```
hive> select date_temp, temperature from temperature_data where zip_code between '300000' and '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
Time taken: 1.317 seconds, Fetched: 6 row(s)
```

2. Calculate maximum temperature corresponding to every year from temperature_data table.

OUTPUT

```
hive> Select substr(date_temp,7), max(temperature) as maxtemp from temperature_data group by substr(date_temp,7);
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different
Query ID = acadgild_20180807003800_d16e8fa7-f31d-4eb2-9832-81233cdec92d
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_1533574160788_0024, Tracking URL = http://localhost:8088/proxy/application_1533574160788_0024/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1533574160788_0024
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2018-08-07 00:38:12,634 Stage-1 map = 0%,  reduce = 0%
2018-08-07 00:38:23,886 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 3.19 sec
2018-08-07 00:38:41,251 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 5.77 sec
MapReduce Total cumulative CPU time: 5 seconds 770 msec
Ended Job = job_1533574160788_0024
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 5.77 sec   HDFS Read: 9207 HDFS Write: 167 SUCCESS
Total MapReduce CPU Time Spent: 5 seconds 770 msec
OK
1990    23
1991    22
1992    14
1994    24
Time taken: 41.657 seconds, Fetched: 4 row(s)
hive>
```

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

OUTPUT

```
Time taken: 38.255 seconds, Fetched: 4 row(s)
hive> Select substr(date_temp,7), max(temperature) as maxtemp from temperature_data group by
 substr(date_temp,7)
    > having count(substr(date_temp,7))>=2;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions.
 Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180807004426_6bd015d9-11b4-4f76-8853-25e527685391
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_1533574160788_0026, Tracking URL = http://localhost:8088/proxy/applicatio
n_1533574160788_0026/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_15335741
60788_0026
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2018-08-07 00:44:37,935 Stage-1 map = 0%,  reduce = 0%
2018-08-07 00:44:48,093 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 3.23 sec
2018-08-07 00:45:00,469 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 6.88 sec
MapReduce Total cumulative CPU time: 6 seconds 880 msec
Ended Job = job_1533574160788_0026
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 6.88 sec   HDFS Read: 10239 HDFS Write: 1
27 SUCCESS
Total MapReduce CPU Time Spent: 6 seconds 880 msec
OK
1990    23
1991    22
Time taken: 36.451 seconds, Fetched: 2 row(s)
```

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

OUTPUT

```
hive> CREATE VIEW temperature_data_vw as
    > Select substr(date_temp,7), max(temperature) as maxtemp from temperature_data
    > group by substr(date_temp,7)
    > having count(substr(date_temp,7))>=2;
OK
Time taken: 3.852 seconds
```

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

OUTPUT

```
hive> insert overwrite local directory '/home/acadgild/max_temp_data1'
    > 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 futu
re versions. Consider using a different execution engine (i.e. spark, tez) or us
ing Hive 1.X releases.
Query ID = acadgild_20180807212646_08035e8a-a404-4256-9436-9c2a946cdf82
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_1533654392488_0001, Tracking URL = http://localhost:8088/proxy/application_1533654392488_0001/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1533654392488_0001
```

```
[acadgild@localhost ~]$ cd max_temp_data1
[acadgild@localhost max_temp_data1]$ ls
000000_0
[acadgild@localhost max_temp_data1]$ cat 000000_0
1990|23
1991|22
[acadgild@localhost max_temp_data1]$
```

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

This Data set is about Olympics. 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

Creating and Loading data into table

```
hive> CREATE table olympics_info(
    > athlete string,
    > age int,
    > country string,
    > year int,
    > 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: 0.232 seconds
hive> load data local inpath '/home/acadgild/olympix_data.csv' into table olympics_info;
Loading data to table custom.olympics_info
OK
Time taken: 0.587 seconds
```

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

Query:

```
hive> select country, count (total_medals) from olympics_info
    > where sport='Swimming' group by country;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a di
Query ID = acadgild_20180807024830_0efa5213-3358-4f2a-b248-f30b5ac1e4fe
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_1533574160788_0031, Tracking URL = http://localhost:8088/proxy/application_1533574160788_0031/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1533574160788_0031
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2018-08-07 02:48:41,894 Stage-1 map = 0%,  reduce = 0%
2018-08-07 02:48:53,469 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 3.54 sec
2018-08-07 02:49:04,951 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 6.16 sec
MapReduce Total cumulative CPU time: 6 seconds 160 msec
Ended Job = job_1533574160788_0031
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 6.16 sec   HDFS Read: 529092 HDFS Write: 878 SUCCESS
```

OUTPUT

```
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: 36.857 seconds, Fetched: 34 row(s)
hive>
```

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

QUERY:

```
hive> Select year , count (total_medals) from olympics_info
    > Where country = 'India'
    > Group by year;
```

OUTPUT:

```
2000    1
2004    1
2008    3
2012    6
Time taken: 34.739 seconds, Fetched: 4 row(s)
hive>
```

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

QUERY:

```
hive> Select country , sum(total_medals)
    > From Olympics_info
    > Group by country;
```

OUTPUT:

```
Total MapReduce CPU Time Spent: 4 se
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
```

4. Write a Hive program to find the number of gold medals each country won.

QUERY:

```
hive> Select country, sum(gold_medals)
    > From olympics_info
    > group by country;
```

OUTPUT:

```
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
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
```

Task 2:

Write a hive UDF that implements functionality of string concat_ws(string SEP, array). 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.

```
hive> desc json_table;
OK
name                    string                  from deserializer
id                      bigint                  from deserializer
skills                  array<string>           from deserializer
Time taken: 0.171 seconds, Fetched: 3 row(s)
hive> select * from json_table;
OK
Amit    1       ["Hadoop","Python"]
Sumit   2       ["Hadoop","Hive"]
Rohit   3       ["Oozie","Python"]
Time taken: 0.309 seconds, Fetched: 3 row(s)
```

Adding jar file and creating a temporary function concat.

```
hive> ADD JAR /home/acadgild/hive_concat.jar;
Added [/home/acadgild/hive_concat.jar] to class path
Added resources: [/home/acadgild/hive_concat.jar]
hive> CREATE TEMPORARY FUNCTION concat AS 'hive_udf.concatenate_udf';
OK
Time taken: 0.01 seconds
hive> select concat ('|',skills) from json_table;
OK
Hadoop|Python
Hadoop|Hive
Oozie|Python
Time taken: 0.309 seconds, Fetched: 3 row(s)
hive>
```

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

STEP1: 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: 0.184 seconds
hive>
```

STEP 3: Inserting Data

```
hive> INSERT INTO table college values(1,'nec','nlr'),(2,'vit','vlr'),(3,'srm','che
n'),(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 future versions. Consider usi
Query ID = acadgild_20180807033410_1e7cf91c-def0-46ae-9800-e7e74f6061d9
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_1533574160788_0037, Tracking URL = http://localhost:8088/proxy/application_153357416078
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1533574160788_0037
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 5
2018-08-07 03:34:26,492 Stage-1 map = 0%,  reduce = 0%
2018-08-07 03:34:42,786 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 4.21 sec
2018-08-07 03:35:28,555 Stage-1 map = 100%,  reduce = 13%, Cumulative CPU 5.23 sec
2018-08-07 03:35:30,147 Stage-1 map = 100%,  reduce = 27%, Cumulative CPU 6.33 sec
2018-08-07 03:35:31,598 Stage-1 map = 100%,  reduce = 53%, Cumulative CPU 9.35 sec
2018-08-07 03:35:35,958 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 12.24 sec
2018-08-07 03:35:55,955 Stage-1 map = 100%,  reduce = 80%, Cumulative CPU 23.92 sec
2018-08-07 03:35:57,358 Stage-1 map = 100%,  reduce = 86%, Cumulative CPU 27.58 sec
2018-08-07 03:35:59,018 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 33.35 sec
MapReduce Total cumulative CPU time: 33 seconds 540 msec
Ended Job = job_1533574160788_0037
Loading data to table custom.college
```

```
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: 0.387 seconds, Fetched: 7 row(s)
hive>
```

STEP 4: Updating the Data in Hive Table

Updating of Bucketing Columns: Results in error as updating of values of bucketing column is not supported.

```
hive> UPDATE college set clg_id = 8 where clg_id = 7;
FAILED: SemanticException [Error 10302]: Updating values of bucketing columns is not supported.  Column clg_id.
```

Update operation on Non bucketed column

```
hive> UPDATE college set clg_name = 'IIT' where clg_id = 6;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future ver
Query ID = acadgild_20180807034246_70761871-a443-4953-8cb6-6368dc3003b4
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_1533574160788_0038, Tracking URL = http://localhost:8088/proxy/appl
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_15
Hadoop job information for Stage-1: number of mappers: 5; number of reducers: 5
2018-08-07 03:43:03,236 Stage-1 map = 0%,  reduce = 0%
2018-08-07 03:44:03,853 Stage-1 map = 0%,  reduce = 0%, Cumulative CPU 13.16 sec
2018-08-07 03:44:15,808 Stage-1 map = 60%,  reduce = 0%, Cumulative CPU 20.17 sec
2018-08-07 03:44:18,830 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 25.22 sec
2018-08-07 03:44:59,172 Stage-1 map = 100%,  reduce = 13%, Cumulative CPU 26.28 sec
2018-08-07 03:45:02,154 Stage-1 map = 100%,  reduce = 27%, Cumulative CPU 27.79 sec
2018-08-07 03:45:04,920 Stage-1 map = 100%,  reduce = 53%, Cumulative CPU 31.06 sec
2018-08-07 03:45:07,912 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 33.37 sec
2018-08-07 03:45:12,173 Stage-1 map = 100%,  reduce = 73%, Cumulative CPU 34.95 sec
2018-08-07 03:45:13,549 Stage-1 map = 100%,  reduce = 80%, Cumulative CPU 36.34 sec
2018-08-07 03:45:14,928 Stage-1 map = 100%,  reduce = 87%, Cumulative CPU 38.73 sec
2018-08-07 03:45:16,150 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 42.13 sec
MapReduce Total cumulative CPU time: 42 seconds 130 msec
Ended Job = job_1533574160788_0038
Loading data to table custom.college
```

```
hive> select * from college;
OK
5       stanford        uk
6       IIT     atp
1       nec     nlr
7       cambridge       us
2       vit     vlr
3       srm     chen
4       lpu     del
Time taken: 0.612 seconds, Fetched: 7 row(s)
```

STEP 5: Deleting a Row from Hive Table

```
hive> delete from college where clg_id=5;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a dif
Query ID = acadgild_20180807035326_062609d0-cbd2-4c49-a5cc-31bcc5ea7bde
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_1533574160788_0039, Tracking URL = http://localhost:8088/proxy/application_1533574160788_0039/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1533574160788_0039
Hadoop job information for Stage-1: number of mappers: 5; number of reducers: 5
2018-08-07 03:53:42,230 Stage-1 map = 0%,  reduce = 0%
2018-08-07 03:54:53,483 Stage-1 map = 60%,  reduce = 0%, Cumulative CPU 16.61 sec
2018-08-07 03:54:55,942 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 25.19 sec
2018-08-07 03:55:42,713 Stage-1 map = 100%,  reduce = 27%, Cumulative CPU 27.21 sec
2018-08-07 03:55:45,516 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 32.78 sec
2018-08-07 03:55:54,126 Stage-1 map = 100%,  reduce = 73%, Cumulative CPU 34.26 sec
2018-08-07 03:55:55,224 Stage-1 map = 100%,  reduce = 93%, Cumulative CPU 39.29 sec
2018-08-07 03:55:56,280 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 42.08 sec
MapReduce Total cumulative CPU time: 42 seconds 80 msec
Ended Job = job_1533574160788_0039
Loading data to table custom.college
MapReduce Jobs Launched:
Stage-Stage-1: Map: 5  Reduce: 5   Cumulative CPU: 42.08 sec   HDFS Read: 50263 HDFS Write: 747 SUCCESS
Total MapReduce CPU Time Spent: 42 seconds 80 msec
OK
Time taken: 155.945 seconds
```

OUTPUT

```
hive> select * from college;
OK
6       IIT     atp
1       nec     nlr
7       cambridge       us
2       vit     vlr
3       srm     chen
4       lpu     del
Time taken: 1.063 seconds, Fetched: 6 row(s)
```

Number 5 deleted from the table