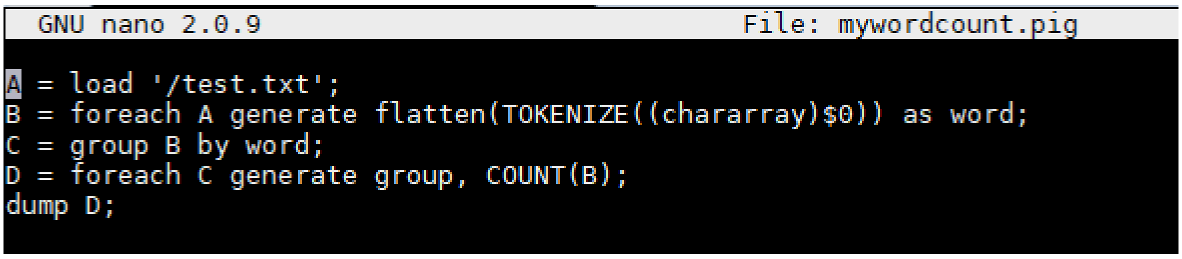
Assignment - 7.1

Task1: Write a program to implement wordcount using Pig.

Answer1: Here the word count code will be executed using a Pig file – **mywordcount.pig**.



```
  GNU nano 2.0.9                           File: mywordcount.pig

A = load '/test.txt';
B = foreach A generate flatten(TOKENIZE((chararray)$0)) as word;
C = group B by word;
D = foreach C generate group, COUNT(B);
dump D;
```

Fig 7.1.1 mywordcount.pig

Relation A ➔ Loads the data from HDFS location.

Relation B ➔ Converts the sentence into array elements.

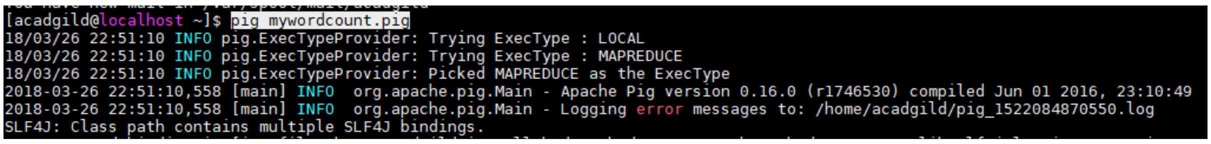
Relation C ➔ Groups B by particular words.

Relation D ➔ Gets the count for each of the Grouped items in C.

Dump command is used to see the output on screen(refer Fig 7.1.3).

Input command:

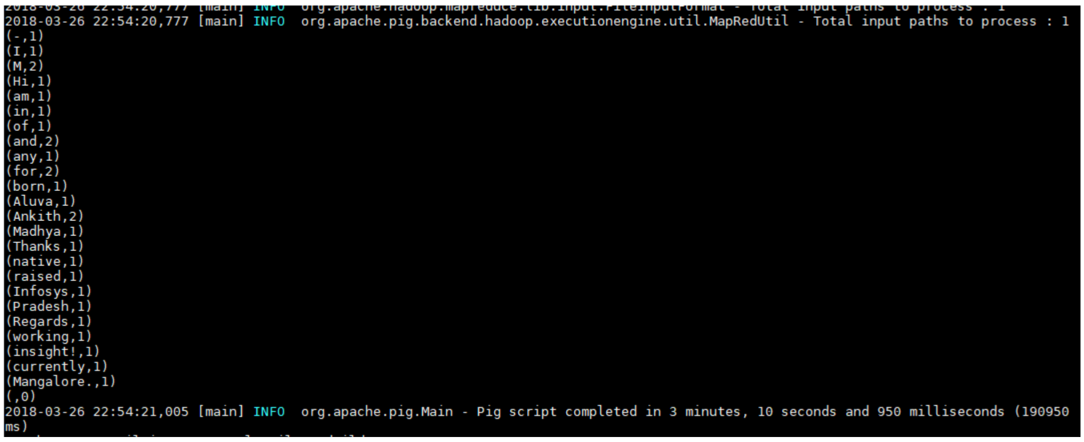
pig mywordcount.pig



```
[acadgild@localhost ~]$ pig mywordcount.pig
18/03/26 22:51:10 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL
18/03/26 22:51:10 INFO pig.ExecTypeProvider: Trying ExecType : MAPREDUCE
18/03/26 22:51:10 INFO pig.ExecTypeProvider: Picked MAPREDUCE as the ExecType
2018-03-26 22:51:10,558 [main] INFO  org.apache.pig.Main - Apache Pig version 0.16.0 (r1746530) compiled Jun 01 2016, 23:10:49
2018-03-26 22:51:10,558 [main] INFO  org.apache.pig.Main - Logging error messages to: /home/acadgild/pig_1522084870550.log
SLF4J: Class path contains multiple SLF4J bindings.
```

Fig 7.1.2 Input command

Output:



```
2018-03-26 22:54:20,777 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(-,1)
(I,1)
(M,2)
(Hi,1)
(am,1)
(in,1)
(of,1)
(and,2)
(any,1)
(for,2)
(born,1)
(Aluva,1)
(Ankith,2)
(Madhya,1)
(Thanks,1)
(native,1)
(raised,1)
(Infosys,1)
(Pradesh,1)
(Regards,1)
(working,1)
(insight!,1)
(currently,1)
(Mangalore.,1)
(,0)
2018-03-26 22:54:21,005 [main] INFO  org.apache.pig.Main - Pig script completed in 3 minutes, 10 seconds and 950 milliseconds (190950
ms)
```

Fig 7.1.3 mywordcount - Output

Task2: We have employee_details and employee_expenses files. Use local mode while running Pig and write Pig Latin script to get below results:

employee_details (EmpID,Name,Salary,Rating)

employee_expenses(EmpID,Expence)

Load the files using below commands:

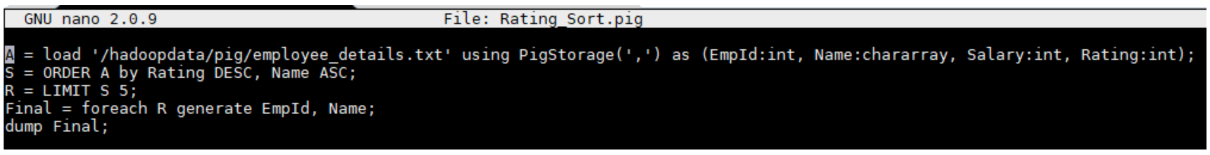
A = load '/hadoopdata/pig/employee_details.txt' using PigStorage(',') as (EmpId:int, Name:chararray, Salary:int, Rating:int);

B = load '/hadoopdata/pig/employee_expenses.txt' using PigStorage('\t') as (EmpId:int, Expense:int);

Task2/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)

Answer:

All the pig operations are written in Pig file as mentioned below (refer Fig 7.2.1). Execution of pig file is done to get the final output.



```
  GNU nano 2.0.9                           File: Rating_Sort.pig

A = load '/hadoopdata/pig/employee_details.txt' using PigStorage(',') as (EmpId:int, Name:chararray, Salary:int, Rating:int);
S = ORDER A by Rating DESC, Name ASC;
R = LIMIT S 5;
Final = foreach R generate EmpId, Name;
dump Final;
```

Fig 7.2.1

Please find the explanation of the pig script as follows.

Relation A➔Loads employee_details.txt file.

Relation S ➔Sorts the file on descending order of Rating and ascending order of Name.

Relation R ➔Limit output by first 5 rows only.

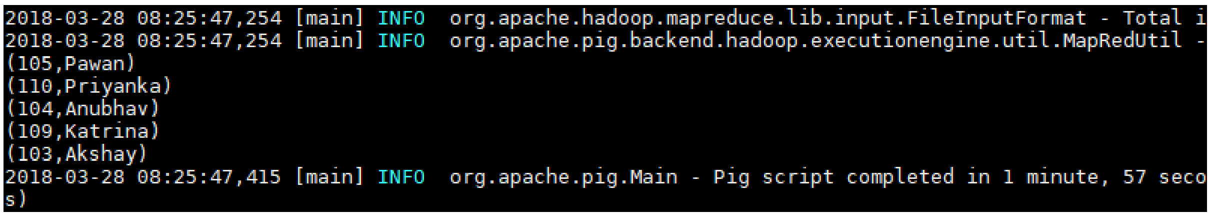
Relation Final ➔ Gets the final output format – EmpID and Name.

Dump command is used to see the output generated (refer Fig 7.2.2).

Input:

pig Rating_Sort.pig

Output:



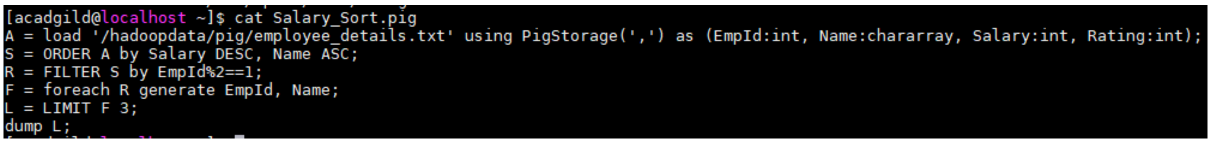
```
2018-03-28 08:25:47,254 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total i
2018-03-28 08:25:47,254 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil -
(105,Pawan)
(110,Priyanka)
(104,Anubhav)
(109,Katrina)
(103,Akshay)
2018-03-28 08:25:47,415 [main] INFO  org.apache.pig.Main - Pig script completed in 1 minute, 57 seco
s)
```

Fig 7.2.2

Task2/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).

Answer:

All the pig operations are written in Pig file as mentioned below. Execution of pig file is done to get the final output.



```
[acadgild@localhost ~]$ cat Salary_Sort.pig
A = load '/hadoopdata/pig/employee_details.txt' using PigStorage(',') as (EmpId:int, Name:chararray, Salary:int, Rating:int);
S = ORDER A by Salary DESC, Name ASC;
R = FILTER S by EmpId%2==1;
F = foreach R generate EmpId, Name;
L = LIMIT F 3;
dump L;
```

Please find the explanation of the pig script as follows.

Relation A➔Loads employee_details.txt file.

Relation S ➔Sorts the file on descending order of Salary and ascending order of Name.

Relation R ➔filters the file to get records with odd EmpId.

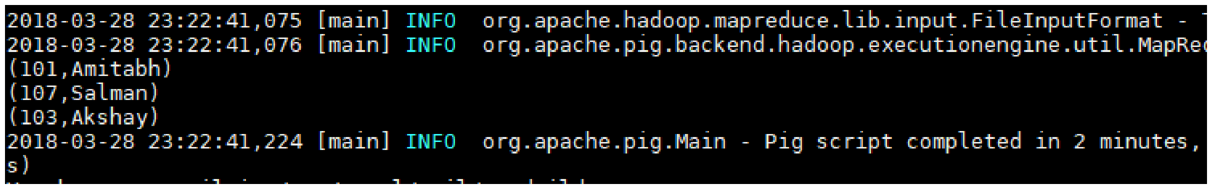
Relation F ➔generates the final template.

Relation L ➔ Limits F to get the top three records only.

Input:

pig Salary_Sort.pig

Output:

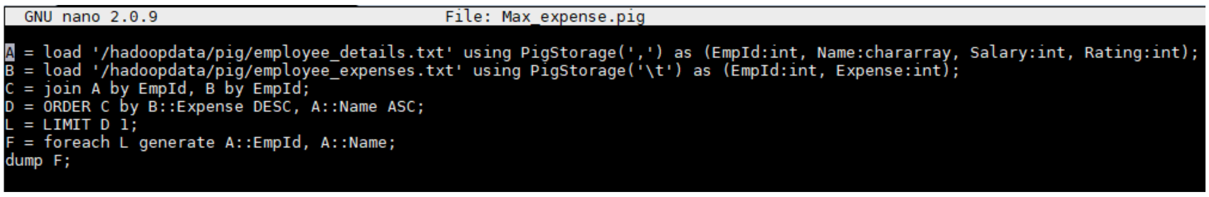


```
2018-03-28 23:22:41,075 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - 
2018-03-28 23:22:41,076 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRed
(101,Amitabh)
(107,Salman)
(103,Akshay)
2018-03-28 23:22:41,224 [main] INFO  org.apache.pig.Main - Pig script completed in 2 minutes,
s)
```

Task2/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)

Answer:

All the pig operations are written in Pig file as mentioned below. Execution of pig file is done to get the final output.



```
  GNU nano 2.0.9                        File: Max_expense.pig

A = load '/hadoopdata/pig/employee_details.txt' using PigStorage(',') as (EmpId:int, Name:chararray, Salary:int, Rating:int);
B = load '/hadoopdata/pig/employee_expenses.txt' using PigStorage('\t') as (EmpId:int, Expense:int);
C = join A by EmpId, B by EmpId;
D = ORDER C by B::Expense DESC, A::Name ASC;
L = LIMIT D 1;
F = foreach L generate A::EmpId, A::Name;
dump F;
```

Please find the explanation of the pig script as follows.

Relation A ➔Loads employee_details.txt file.

Relation B ➔ Loads employee_expenses.txt file.

Relation C ➔Joins both files on Employee ID.

Relation D ➔ Sorts the file on descending order of expenses and ascending order of Name.

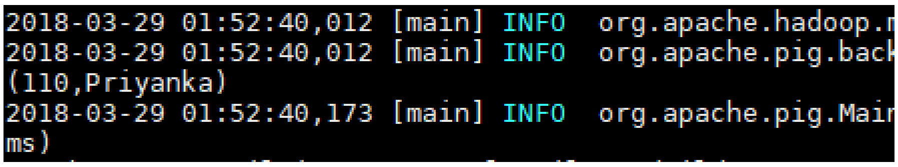
Relation L ➔ Limits D to get only one record.

Relation F ➔generates the final template.

Input:

pig Max_expense.pig

Output:

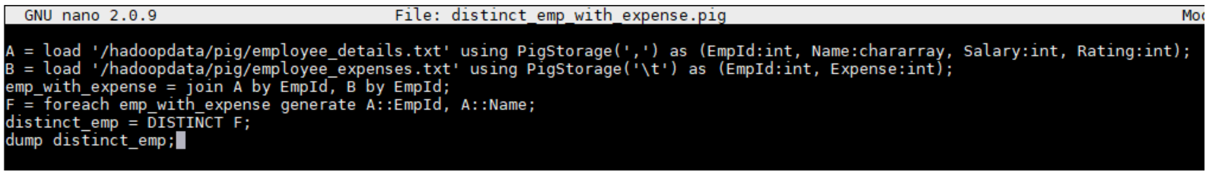


```
2018-03-29 01:52:40,012 [main] INFO  org.apache.hadoop.r
2018-03-29 01:52:40,012 [main] INFO  org.apache.pig.back
(110,Priyanka)
2018-03-29 01:52:40,173 [main] INFO  org.apache.pig.Main
ms)
```

Task2/d: List of employees (employee id and employee name) having entries in employee_expenses file.

Answer:

All the pig operations are written in Pig file as mentioned below. Execution of pig file is done to get the final output.



```
GNU nano 2.0.9                    File: distinct_emp_with_expense.pig                                    Mo
A = load '/hadoopdata/pig/employee_details.txt' using PigStorage(',') as (EmpId:int, Name:chararray, Salary:int, Rating:int);
B = load '/hadoopdata/pig/employee_expenses.txt' using PigStorage('\t') as (EmpId:int, Expense:int);
emp_with_expense = join A by EmpId, B by EmpId;
F = foreach emp_with_expense generate A::EmpId, A::Name;
distinct_emp = DISTINCT F;
dump distinct_emp;
```

Please find the explanation of the pig script as follows.

Relation A ➔Loads employee_details.txt file.

Relation B ➔ Loads employee_expenses.txt file.

Relation **emp_with_expense** ➔Joins both files on Employee ID.

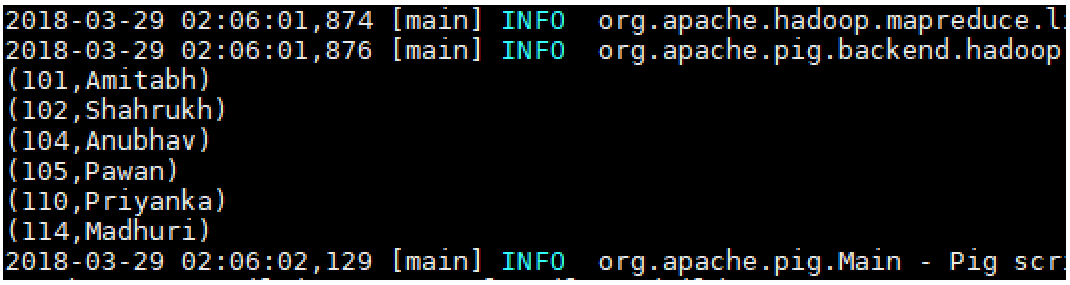
Relation F ➔generates the final template.

Relation **distinct_emp** ➔generates unique set of employees by removing duplicates.

Input:

pig distinct_emp_with_expense.pig

Output:

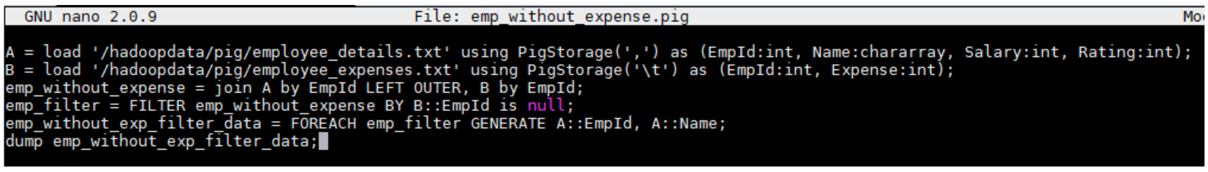


```
2018-03-29 02:06:01,874 [main] INFO  org.apache.hadoop.mapreduce.l
2018-03-29 02:06:01,876 [main] INFO  org.apache.pig.backend.hadoop
(101,Amitabh)
(102,Shahrukh)
(104,Anubhav)
(105,Pawan)
(110,Priyanka)
(114,Madhuri)
2018-03-29 02:06:02,129 [main] INFO  org.apache.pig.Main - Pig scr
```

Task2/e: List of employees (employee id and employee name) having no entry in employee_expenses file.

Answer:

All the pig operations are written in Pig file as mentioned below. Execution of pig file is done to get the final output.



```
GNU nano 2.0.9                    File: emp_without_expense.pig                                          Mo
A = load '/hadoopdata/pig/employee_details.txt' using PigStorage(',') as (EmpId:int, Name:chararray, Salary:int, Rating:int);
B = load '/hadoopdata/pig/employee_expenses.txt' using PigStorage('\t') as (EmpId:int, Expense:int);
emp_without_expense = join A by EmpId LEFT OUTER, B by EmpId;
emp_filter = FILTER emp_without_expense BY B::EmpId is null;
emp_without_exp_filter_data = FOREACH emp_filter GENERATE A::EmpId, A::Name;
dump emp_without_exp_filter_data;
```

Please find the explanation of the pig script as follows.

Relation A ➔Loads employee_details.txt file.

Relation B ➔ Loads employee_expenses.txt file.

Relation **emp_without_expense** ➔Joins both files on Employee ID using LEFT OUTER join.

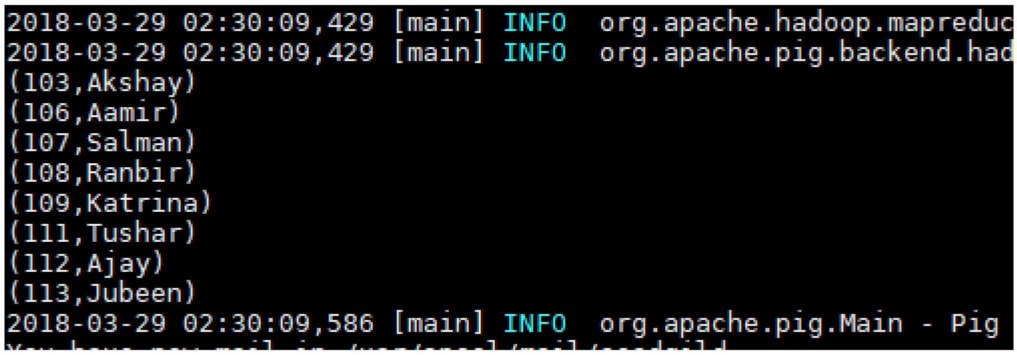
Relation emp_filter ➔Previous output is filtered only for the records with no EmpId in expenses file.

Relation **emp_without_exp_filter_data** ➔gets the final output format.

Input:

pig emp_without_expense.pig

Output:



```
2018-03-29 02:30:09,429 [main] INFO  org.apache.hadoop.mapreduc
2018-03-29 02:30:09,429 [main] INFO  org.apache.pig.backend.had
(103,Akshay)
(106,Aamir)
(107,Salman)
(108,Ranbir)
(109,Katrina)
(111,Tushar)
(112,Ajay)
(113,Jubeen)
2018-03-29 02:30:09,586 [main] INFO  org.apache.pig.Main - Pig
```

Task 3 Prob1

Task3: Implement the use case present in below blog link and share the complete steps along with screenshot(s) from your end.

Problem1: Find out the top 5 most visited destinations.

Answer:

Input commands:

REGISTER '/home/acadgild/piggybank.jar';

A = load '/aviationdata/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;

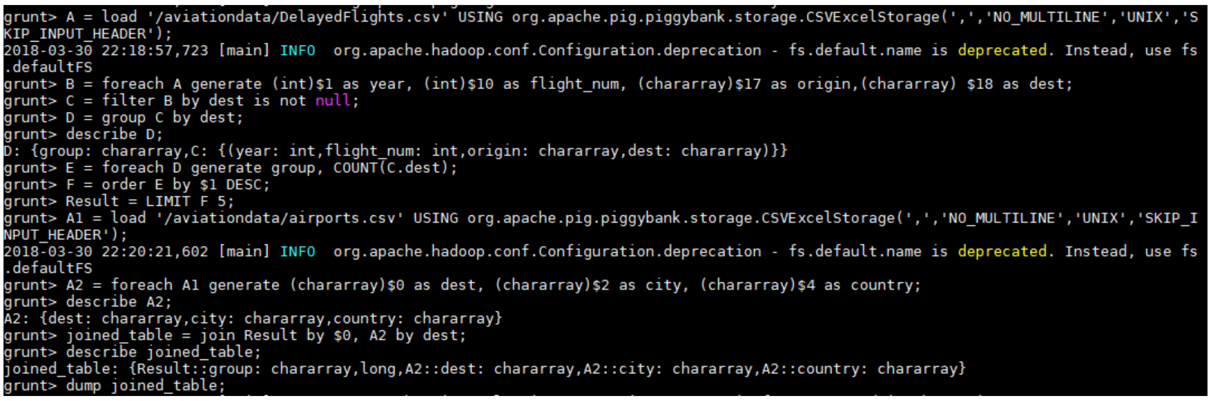
Result = LIMIT F 5;

A1 = load '/aviationdata/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;

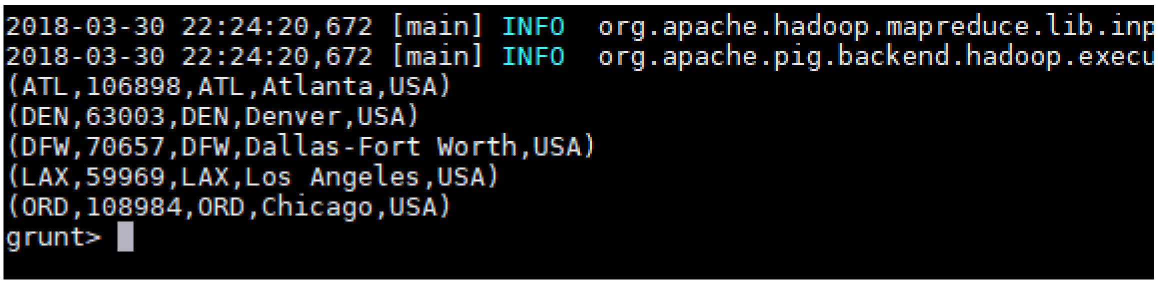
joined_table = join Result by $0, A2 by dest;

dump joined_table;



```
grunt> A = load '/aviationdata/DelayedFlights.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',','NO_MULTILINE','UNIX','S
KIP_INPUT_HEADER');
2018-03-30 22:18:57,723 [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> describe D;
D: {group: chararray,C: {(year: int,flight_num: int,origin: chararray,dest: chararray)}}
grunt> E = foreach D generate group, COUNT(C.dest);
grunt> F = order E by $1 DESC;
grunt> Result = LIMIT F 5;
grunt> A1 = load '/aviationdata/airports.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',','NO_MULTILINE','UNIX','SKIP_I
NPUT_HEADER');
2018-03-30 22:20:21,602 [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> describe A2;
A2: {dest: chararray,city: chararray,country: chararray}
grunt> joined_table = join Result by $0, A2 by dest;
grunt> describe joined_table;
joined_table: {Result::group: chararray,long,A2::dest: chararray,A2::city: chararray,A2::country: chararray}
grunt> dump joined_table;
```

Output:



```
2018-03-30 22:24:20,672 [main] INFO  org.apache.hadoop.mapreduce.lib.inp
2018-03-30 22:24:20,672 [main] INFO  org.apache.pig.backend.hadoop.execu
(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?

Answer:

Input commands:

A = load '/aviationdata/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';

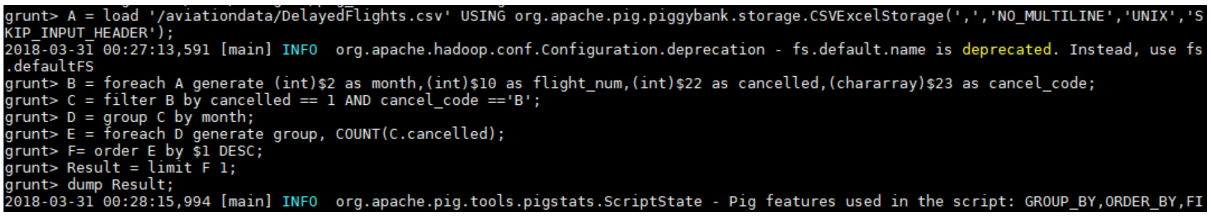
D = group C by month;

E = foreach D generate group, COUNT(C.cancelled);

F= order E by $1 DESC;

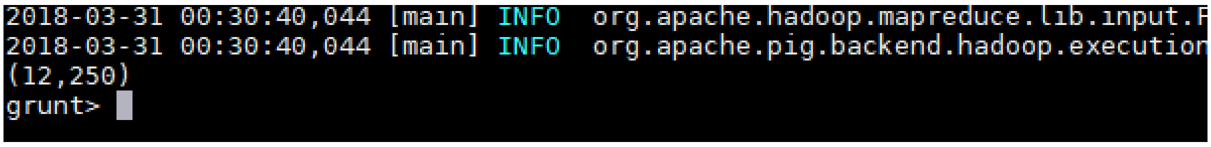
Result = limit F 1;

dump Result;



```
grunt> A = load '/aviationdata/DelayedFlights.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',','NO_MULTILINE','UNIX','S
KIP_INPUT_HEADER');
2018-03-31 00:27:13,591 [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;
2018-03-31 00:28:15,994 [main] INFO  org.apache.pig.tools.pigstats.ScriptState - Pig features used in the script: GROUP_BY,ORDER_BY,FI
```

Output:



```
2018-03-31 00:30:40,044 [main] INFO  org.apache.hadoop.mapreduce.lib.input.F
2018-03-31 00:30:40,044 [main] INFO  org.apache.pig.backend.hadoop.execution
(12,250)
grunt>
```

Problem3: Top ten origins with the highest AVG departure delay.

Answer:

Input commands:

A = load '/aviationdata/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 '/aviationdata/airports.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',','NO_MULTILINE','UNIX','SKIP_INPUT_HEADER')
;

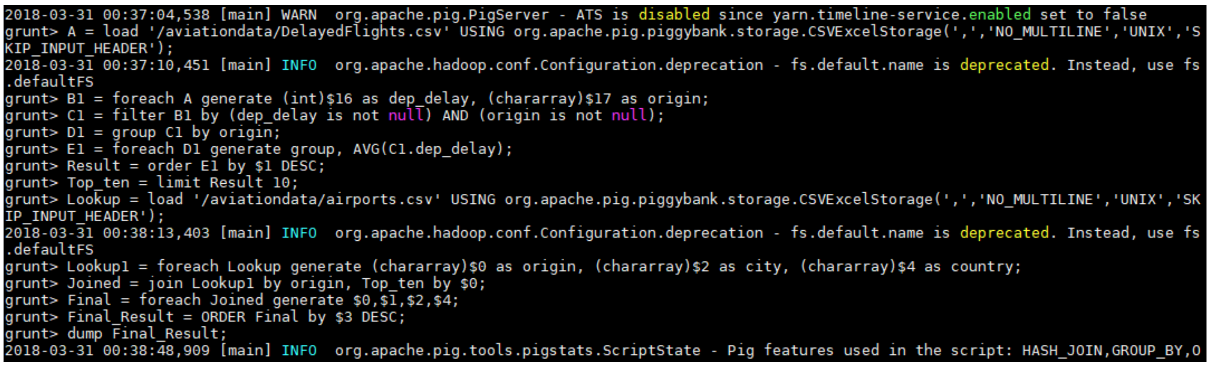
Lookup1 = foreach Lookup generate (chararray)$0 as origin, (chararray)$2 as city, (chararray)$4 as country;

Joined = join Lookup1 by origin, Top_ten by $0;

Final = foreach Joined generate $0,$1,$2,$4;

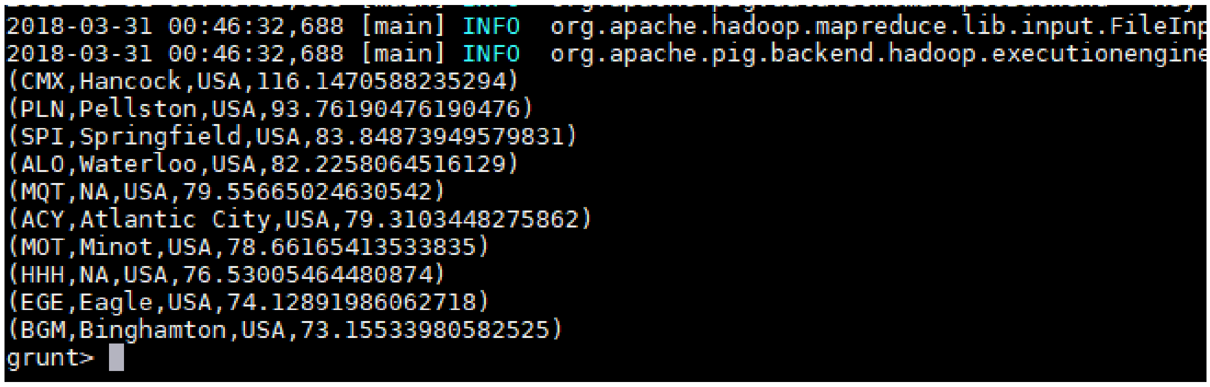
Final_Result = ORDER Final by $3 DESC;

dump Final_Result;



```
2018-03-31 00:37:04,538 [main] WARN  org.apache.pig.PigServer - ATS is disabled since yarn.timeline-service.enabled set to false
grunt> A = load '/aviationdata/DelayedFlights.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',','NO_MULTILINE','UNIX','S
KIP_INPUT_HEADER');
2018-03-31 00:37:10,451 [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 '/aviationdata/airports.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',','NO_MULTILINE','UNIX','SK
IP_INPUT_HEADER');
2018-03-31 00:38:13,403 [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;
2018-03-31 00:38:48,909 [main] INFO  org.apache.pig.tools.pigstats.ScriptState - Pig features used in the script: HASH_JOIN,GROUP_BY,O
```

Output:



```
2018-03-31 00:46:32,688 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInp
2018-03-31 00:46:32,688 [main] INFO  org.apache.pig.backend.hadoop.executionengine
(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>
```

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

Answer:

Input commands:

A = load '/aviationdata/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);

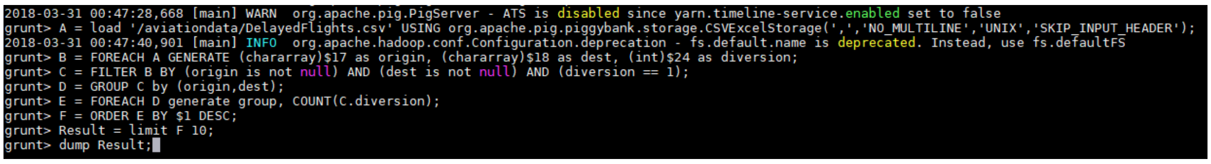
D = GROUP C by (origin,dest);

E = FOREACH D generate group, COUNT(C.diversion);

F = ORDER E BY $1 DESC;

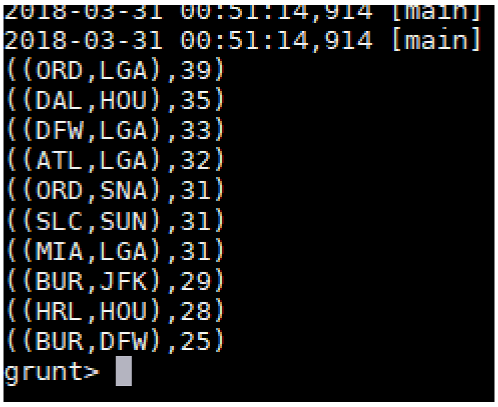
Result = limit F 10;

dump Result;



```
2018-03-31 00:47:28,668 [main] WARN  org.apache.pig.PigServer - ATS is disabled since yarn.timeline-service.enabled set to false
grunt> A = load '/aviationdata/DelayedFlights.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',','NO_MULTILINE','UNIX','SKIP_INPUT_HEADER');
2018-03-31 00:47:40,901 [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;
```

Output:



```
2018-03-31 00:51:14,914 [main]
2018-03-31 00:51:14,914 [main]
((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>
```