ASSIGNMENT 7

EXPLORING APACHE PIG

PROBLEM STATEMENT –

Write PIG scripts for following tasks.

Load Data into Pig Storage:

Before going into the tasks we are loading the data sets to Pig Storage.

employee_details = LOAD ' /employee_details.txt' USING PigStorage(',') AS (id:int, name:chararray, salary:int, rating:int);

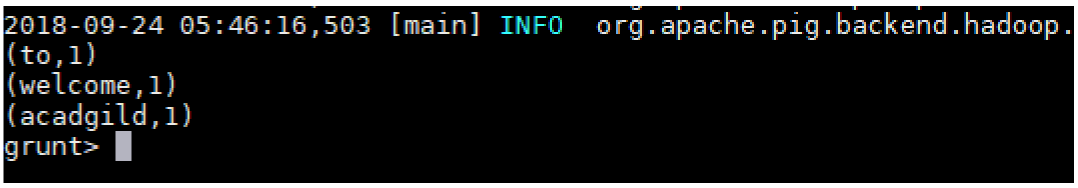
employee_expenses = LOAD ' /employee_expenses.txt' USING PigStorage() AS (id:int, expenses:int);

Task 1

Write a program to implement wordcount using Pig.

We are loading the file into relation A, and tokenize each word and store in relation B. Group the words in relation C and generate the count of each word and store in relation D.

1. **A = load '/home/acadgild/hadoop/test.txt';**
2. **B = foreach A generate flatten(TOKENIZE((chararray)$0)) as word;**
3. **C = group B by word;**
4. **D = foreach C generate group, COUNT(B);**
5. **dump D;**



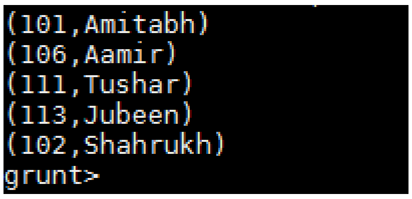
```
2018-09-24 05:46:16,503 [main] INFO  org.apache.pig.backend.hadoop.
(to,1)
(welcome,1)
(acadgild,1)
grunt>
```

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)

We are using the function ORDER to order the rating and the ID by ascending order and we are limiting the employee by top 5 with id and the name.

1. **employee = ORDER employee_details BY rating asc, id asc;**
2. **limit_employee = LIMIT employee 5;**
3. **top_employee = FOREACH limit_employee GENERATE id,name;**

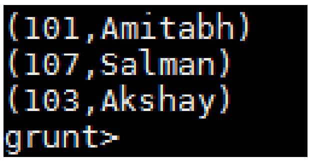


```
(101,Amitabh)
(106,Aamir)
(111,Tushar)
(113,Jubeen)
(102,Shahrukh)
grunt>
```

(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)

We are using the function ORDER to order the employee_details by arranging the salary in descending order and filtering the even number using the logic **(id%2)!=0** and limiting the employee by 3 and bringing the result according to the employee ID and Name.

1. **order_employee = ORDER employee_details BY salary DESC;**
2. **filter_employee = FILTER order_employee BY (id%2)!=0;**
3. **limit_employee = LIMIT filter_employee 3;**
4. **top_3employee = FOREACH limit_employee GENERATE id,name;**
5. **DUMP top_3employee;**

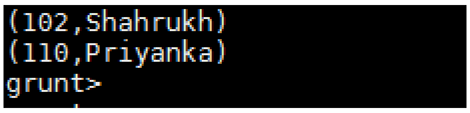


```
(101,Amitabh)
(107,Salman)
(103,Akshay)
grunt>
```

(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)

In this task, we are joining the tables employee_details and employee_expenses using the JOIN function and sorting the expenses by Descending order and limiting the employee first 2 rows and bringing the id and name of the employee as the output.

1. **emp_det_exp = JOIN employee_details BY id, emp_expenses by id;**
2. **order_exp = ORDER emp_det_exp BY expenses DESC;**
3. **limit_exp = LIMIT order_exp 2;**
4. **max_exp = FOREACH limit_exp GENERATE $0,$1;**
5. **DUMP max_exp;**

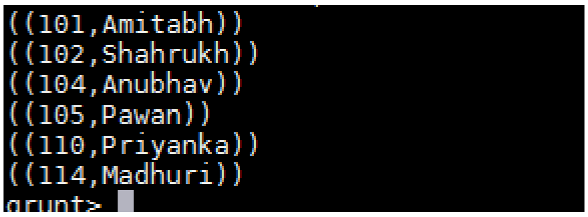


```
(102,Shahrukh)
(110,Priyanka)
grunt>
```

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

We are joining the table's employee_details and employee_expenses using the JOIN function and removing the redundant tuples from the relation and displaying the ID and name of the employee.

1. **emp_det_exp = JOIN employee_details BY id, emp_expenses by id;**
2. **emp_names = FOREACH emp_det_exp GENERATE($0,$1);**
3. **names = DISTINCT emp_names;**
4. **DUMP names;**

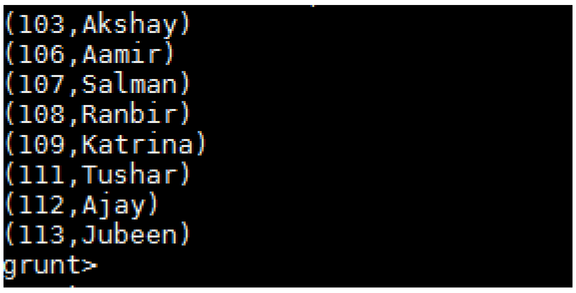


```
((101,Amitabh))
((102,Shahrukh))
((104,Anubhav))
((105,Pawan))
((110,Priyanka))
((114,Madhuri))
grunt>
```

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

Using the LEFT OUTER JOIN, we are joining the table's employee_details and employee_expenses and filtering the empty values in the columns ($4: employee_expenses.id and $5: employee_expenses.expenses) and generating the output by ID and the name of the employee.

1. **emp_det_exp = JOIN employee_details BY id LEFT OUTER, employee_expenses BY id;**
2. **filter_emp_det_exp = FILTER emp_det_exp BY $4 Is NULL and $5 Is NULL;**
3. **gen_emp_det_exp = FOREACH filter_emp_det_exp GENERATE $0,$1;**
4. **DUMP gen_emp_det_exp;**

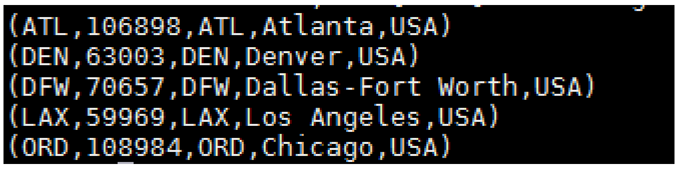

```
(103,Akshay)
(106,Aamir)
(107,Salman)
(108,Ranbir)
(109,Katrina)
(111,Tushar)
(112,Ajay)
(113,Jubeen)
grunt>
```

Task 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/

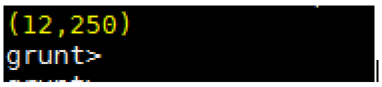
(a) Find out the top 5 most visited destinations

1. REGISTER '/home/acadgild/hadoop/piggybank-0.15.0.jar';
2. A = load '/home/acadgild/hadoop/DelayedFlights.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',','NO_MULTILINE','UNIX','SKIP_INPUT_HEADER');
3. B = FOREACH A GENERATE (int)$1 as year, (int)$10 as flight_num, (chararray)$17 as origin,(chararray) $18 as dest;
4. C = FILTER B by dest is NOT Null;
5. D = GROUP C BY dest;
6. E = FOREACH D GENERATE group, COUNT(C.dest);
7. F = ORDER E by $1 DESC;
8. Result = LIMIT F 5;
9. A1 = LOAD '/home/acadgild/hadoop/airports.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',','NO_MULTILINE','UNIX','SKIP_INPUT_HEADER');
10. A2 = FOREACH A1 GENERATE (chararray)$0 as dest, (chararray)$2 as city, (chararray)$4 as country;
11. joined_table = JOIN Result by $0, A2 by dest;
12. DUMP joined_table;


```
(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)
```

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

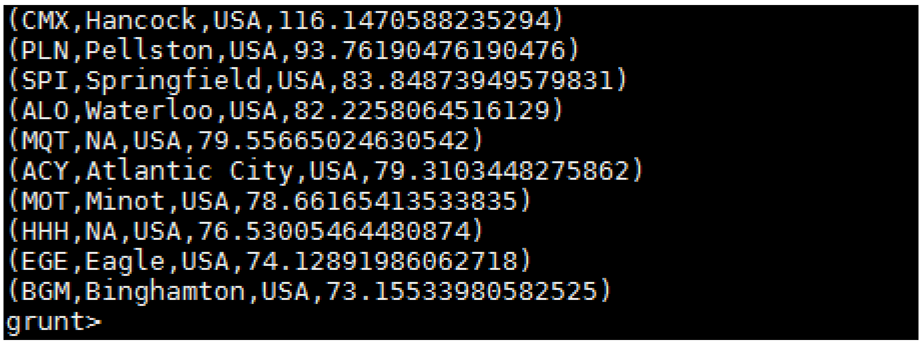
1. REGISTER '/home/acadgild/hadoop/piggybank-0.15.0.jar';
2. A = load '/home/acadgild/hadoop/DelayedFlights.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',','NO_MULTILINE','UNIX','SKIP_INPUT_HEADER');
3. B = FOREACH A GENERATE(int)$2 as month,(int)$10 as flight_num,(int)$22 as cancelled,(chararray)$23 as cancel_code;
4. C = FILTER B by cancelled == 1 AND cancel_code =='B';
5. D = group C BY month;
6. E = FOREACH D GENERATE group, COUNT(C.cancelled);
7. F= ORDER E BY $1 DESC;
8. Result = limit F 1;
9. DUMP Result;



```
(12,250)
grunt>
```

(c) Top ten origins with the highest AVG departure delay

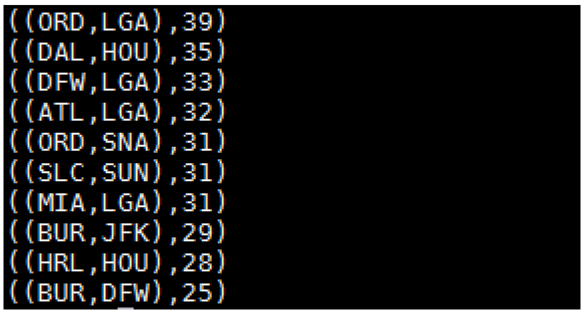
1. REGISTER '/home/acadgild/hadoop/piggybank-0.15.0.jar';
2. A = load '/home/acadgild/hadoop/DelayedFlights.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',','NO_MULTILINE','UNIX','SKIP_INPUT_HEADER');
3. B1 = FOREACH A GENERATE (int)$16 as dep_delay, (chararray)$17 as origin;
4. C1 = FILTER B1 BY (dep_delay is not null) AND (origin is not null);
5. D1 = group C1 by origin;
6. E1 = FOREACH D1 generate group, AVG(C1.dep_delay);
7. Result = ORDER E1 by $1 DESC;
8. Top_ten = LIMIT Result 10;
9. Lookup = load '/home/acadgild/hadoop/airports.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',','NO_MULTILINE','UNIX','SKIP_INPUT_HEADER');
10. Lookup1 = FOREACH Lookup GENERATE (chararray)$0 as origin, (chararray)$2 as city, (chararray)$4 as country;
11. Joined = JOIN Lookup1 by origin, Top_ten by $0;
12. Final = FOREACH Joined GENERATE $0,$1,$2,$4;
13. Final_Result = ORDER Final by $3 DESC;
14. DUMP Final_Result;



```
(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>
```

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

1. **REGISTER '/home/acadgild/airline_usecase/piggybank.jar';**
2. **A = load '/home/acadgild/hadoop/DelayedFlights.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',','NO_MULTILINE','UNIX','SKIP_INPUT_HEADER');**
3. **B = FOREACH A GENERATE (chararray)$17 as origin, (chararray)$18 as dest, (int)$24 as diversion;**
4. **C = FILTER B BY (origin is not null) AND (dest is not null) AND (diversion == 1);**
5. **D = GROUP C by (origin,dest);**
6. **E = FOREACH D generate group, COUNT(C.diversion);**
7. **F = ORDER E BY $1 DESC;**
8. **Result = LIMIT F 10;**
9. **DUMP Result;**



```
((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)
```