Session 5

Assignment 2

|  |  |
| --- | --- |
| **Prepared For:** | AcadGild |
|  |  |
| **Document Approval:** | **AcadGild** |
|  |  |
|  |  |
|  |  |
|  |  |
| **Project Title:** | Session 5 – Assignment 3Pokemon |
|  |  |
| **Prepared By:** | Duncan Burgess |
|  |  |
|  | dburgess@duncb.com |
|  |  |
| **Primary Engineer:** | Duncan Burgess |
|  |  |
| **Document Reference:** | **Session 5 – Assignment 2** |
|  |  |
| **Start Date:** | 15/09/2017 |
|  |  |
|  |  |



# 

# Contents

[Contents 2](#_Toc493481420)

[Change History 3](#_Toc493481421)

[1. Problem Statement 4](#_Toc493481422)

[2. Setting up the PIG environment 5](#_Toc493481423)

[3. Solutions 6](#_Toc493481424)

[3.1. Problem Statement 1 6](#_Toc493481425)

[3.2. Problem Statement 2 8](#_Toc493481426)

[3.3. Problem Statement 3 9](#_Toc493481427)

[3.4. Problem Statement 4 11](#_Toc493481428)

# Change History

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Document Revision** | **Date** | **Authored By** | **Authorised By** | **Sections Affected** | **Reason for Change** |
| Rev 01 | 18/09/2017 | Duncan Burgess |  | All | Initial release. |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

# Problem Statement

Implement the use case present in below blog link and share the complete steps along with

screenshot(s) from your end.

NOTE: You must submit a word file containing steps and screenshots.

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

# Setting up the PIG environment

**Two dataset created to implement use case.**

**Airports.csv**

**Delayed\_Flight.csv**

#,Name,Type1,Type2,Total,HP,Attack,Defense,Sp.Atk,Sp.Def,Speed  
001,Bulbasaur,Grass,Poison,318,45,49,49,65,65,45  
002,Ivysaur,Grass,Poison,405,60,62,63,80,80,60  
003,Venusaur,Grass,Poison,525,80,82,83,100,100,80  
003,VenusaurMegaVenusaur,Grass,Poison,625,80,100,123,122,120,80  
004,Charmander,Fire,,309,39,52,43,60,50,65  
005,Charmeleon,Fire,,405,58,64,58,80,65,80  
006,Charizard,Fire,Flying,534,78,84,78,109,85,100  
006,CharizardMegaCharizardX,Fire,Dragon,634,78,130,111,130,85,100  
006,CharizardMegaCharizardY,Fire,Flying,634,78,104,78,159,115,100  
007,Squirtle,Water,,314,44,48,65,50,64,43  
008,Wartortle,Water,,405,59,63,80,65,80,58

**Copy files to hdfs**

$ hadoop fs -copyFromLocal Airports.csv

$ hadoop fs -copyFromLocal Delayed\_Flights.csv

**Start Pig**

pig -x local - *(local mode)*

2017-09-14 14:32:48,658 INFO [main] pig.ExecTypeProvider: Trying ExecType : LOCAL

2017-09-14 14:32:48,658 INFO [main] pig.ExecTypeProvider: Picked LOCAL as the ExecType

you will be presents with the pig shell: grunt>

We need to load the files now: ***Note PigStorage is cap sensitive.***

# Solutions

These are 2 different datasets, i.e., Delayed\_Flights.csv and Airports.csv. Let us understand one at a time.

Delayed\_Flights.csv Datasets

There are 29 columns in this dataset. Some of them have been mentioned below:

•Year: 1987 – 2008

•Month: 1 – 12

•FlightNum: Flight number

•Canceled: Was the flight canceled?

•CancelleationCode: The reason for cancellation.

Airports.csv Datasets

•iata: the international airport abbreviation code

•name of the airport

•city and country in which airport is located.

•lat and long: the latitude and longitude of the airport

Now, using Apache pig, we will try to gain more insights from these datasets.

## Problem Statement 1

Find out the top 5 most visited destinations.

*grunt> REGISTER '/home/acadgild/piggybank.jar';*

*grunt> A = load '/home/acadgild/Documents/DelayedFlights.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',','NO\_MULTILINE','UNIX','SKIP\_INPUT\_HEADER');*

*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> dump E;*

(ABE,1322)

(ABI,838)

(ABQ,12183)

(ABY,319)

(ACK,155)

(ACT,546)

(ACV,1102)

(ACY,38)

(ADK,31)

(ADQ,105)

(AEX,684)

(AGS,728)

(AKN,43)

(ALB,4320)

(ALO,55)

*grunt> F = order E by $1 DESC;*

*grunt> Result = LIMIT F 5;*

*grunt> A1 = load '/home/acadgild/Documents/airports.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',','NO\_MULTILINE','UNIX','SKIP\_INPUT\_HEADER');*

*grunt> A2 = foreach A1 generate (chararray)$0 as dest, (chararray)$2 as city, (chararray)$4 as country;*

*grunt> describe A2;*

*grunt> A2: {dest: chararray,city: chararray,country: chararray}*

*grunt> joined\_table = join Result by $0, A2 by dest;*

*grunt> dump joined\_table;*

**Results**

(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)

**Explination**

In Line 1: We are registering the piggybank jar in order to use the CSVExcelStorage class.

In relation A, we are loading the dataset using CSVExcelStorage because of its effective technique to handle double quotes and headers.

In relation B, we are generating the columns that are required for processing and explicitly typecasting each of them.

In relation C, we are filtering the null values from the “dest” column.

In relation D, we are grouping relation C by “dest.”

In relation E, we are generating the grouped column and the count of each.

Relation F and Result is used to order and limit the result to top 5.

These are the steps to find the top 5 most visited destinations. However, adding few more steps in this process, we will be using another table to find the city name and country as well.

In relation A1, we are loading another table to which we will look-up and find the city as well as the country.

In relation A2, we are generating dest, city, and country from the previous relation.

In relation joined\_table, we are joining Result and A2 based on a common column, i.e., “dest”

Finally, using dump, we are printing the result.

## Problem Statement 2

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

*grunt> REGISTER '/home/acadgild/piggybank.jar';*

*grunt> A = load '/home/acadgild/Documents/DelayedFlights.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',','NO\_MULTILINE','UNIX','SKIP\_INPUT\_HEADER');*

*grunt> B = foreach A generate (int)$2 as month,(int)$10 as flight\_num,(int)$22 as cancelled,(chararray)$23 as cancel\_code;*

*grunt> describe B;*

grunt> B: {month: int,flight\_num: int,cancelled: int,cancel\_code: chararray}

*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;*

**Result**

(12,250)

**Explination**

In Line 1: We are registering piggybank jar in order to use the CSVExcelStorage class.

In relation A, we are loading the dataset using CSVExcelStorage because of its effective technique to handle double quotes and header.

In relation B, we are generating the columns which are required for processing and explicitly typecasting each of them.

In relation C, we are filtering the data based on cancellation and cancellation code, i.e., canceled = 1 means flight have been canceled and cancel\_code = ‘B’ means the reason for cancellation is “weather.” So relation C will point to the data which consists of canceled flights due to bad weather.

In relation D, we are grouping the relation C based on every month.

In relation E, we are finding the count of canceled flights every month.

Relation F and Result is for ordering and finding the top month based on cancellation.

## Problem Statement 3

Top ten origins with the highest AVG departure delay

*grunt> REGISTER '/home/acadgild/piggybank.jar';*

*grunt> A = load '/home/acadgild/Documents/DelayedFlights.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',','NO\_MULTILINE','UNIX','SKIP\_INPUT\_HEADER');*

*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, ROUND\_TO(((double) AVG(C1.dep\_delay)),2);*

*grunt> Result = order E1 by $1 DESC;*

*grunt> Top\_ten = limit Result 10;*

*grunt> Lookup = load '/home/acadgild/Documents/airports.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',','NO\_MULTILINE','UNIX','SKIP\_INPUT\_HEADER');*

*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;*

Original Output

(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)

Modified Result changing *grunt> E1 = foreach D1 generate group, ROUND\_TO(((double) AVG(C1.dep\_delay)),2);*

To output 2 decimal places.

**Result**

(CMX,Hancock,USA,116.15)

(PLN,Pellston,USA,93.76)

(SPI,Springfield,USA,83.85)

(ALO,Waterloo,USA,82.23)

(MQT,NA,USA,79.56)

(ACY,Atlantic City,USA,79.31)

(MOT,Minot,USA,78.66)

(HHH,NA,USA,76.53)

(EGE,Eagle,USA,74.13)

(BGM,Binghamton,USA,73.16)

**Explination**

The first 3 lines are the same as explained in the previous 2 problem statements.

In relation C1, we are removing the null values fields present if any.

In relation D1, we are grouping the data based on column “origin.”

In relation E1, we are finding average delay from each unique origin.

Relations named Result and Top\_ten are ordering the results in descending order and printing the top ten values.

These steps are good enough to find the top ten origins with the highest average departure delay.

However, rather than generating just the code of origin, we will be following a few more steps to find some more details like country and city.

In the relation Lookup, we are loading another table to which we will look up and find the city as well as the country.

In the relation Lookup1, we are generating the destination, city, and country from the previous relation.

In the relation Joined, we are joining relation Top\_ten and Lookup1 based on common a column, i.e., “origin.”

In the relation Final, we are generating required columns from the Joined table.

Finally, we are ordering and printing the results.

## Problem Statement 4

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

*grunt>* REGISTER '/home/acadgild/piggybank.jar';

*grunt>* A = load '/home/acadgild/Documents/DelayedFlights.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',','NO\_MULTILINE','UNIX','SKIP\_INPUT\_HEADER');

*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;

**Results**

((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)

**Explination**

In Line 1: We are registering piggybank jar in order to use CSVExcelStorage class.

In relation A, we are loading the dataset using CSVExcelStorage because of its effective technique to handle double quotes and headers.

In relation B, we are generating the columns which are required for processing and explicitly type-casting each of them.

In relation C, we are filtering the data based on “not null” and diversion =1. This will remove the null records, if any, and give the data corresponding to the diversion taken.

In relation D, we are grouping the data based on origin and destination.

Relation D finds the count of diversion taken per unique origin and destination.

Relations F and Result orders the result and produces top 10 results.