Paul Ycay

500709618

CIND719 Big Data Analytics Tools

Fall 2019, Wednesday 6:30-9:30 pm section

November 06, 2019

Assignment 2

Scalar types (e.g int, long, float, double, chararray, bytearray)

Comments in %

Dataset columns

Variable Names

Shell commands in standard color; *tables italicized*

RELATIONAL OPERATORS IN CAPS

```
-------------------------
-- Dataset & Preparation
-------------------------

1. In this assignment, you will work with datasets from airline industry. The datasets are available in
https://openflights.org/data.html .
Note that the datasets may contain outdated and obsolete data. Use for learning purposes only.
If you download the datasets directly from its source, you may need to clean up first, such as removing double quotes etc.

2. Download the flight.tgz archive.

3. Copy the archive to your Linux VM.

4. Untar the archive in your Linux VM using the tar command. You will need -x and -f options at a minimum. -x option
extracts an archive and -f identifies the archive file you want to extract. See https://www.interserver.net/tips/kb/use-
tar-command-linux-examples for more info.

5. The archive extracts into a new directory named flight. You will find three datasets in it. Visit
https://openflights.org/data.html for explanation of individual datasets.

6. The data is ISO 8859-1 (Latin-1) encoded. The special value \N is used for "NULL" to indicate that no value is
available. You may need to clean the '\N' values from the datasets if you think it's necessary.

-----------------------------
-- Datasets
--
-- 1. airlines.dat
-- 2. airports.dat
-- 3. routes.dat
-----------------------------
```

Loading Dataset:

https://openflights.org/data.html

%first transfer *flight.tgz* into FileZilla then execute the following

[root@sandbox ~] # hadoop fs -put /root/*flight.tgz* /user/lab

[root@sandbox ~] # tar -xvf *flight.tgz* %tar command allows to quickly access a collection of files and places them into a highly compressed archive file commonly called tarball, or tar, gzip, and bzip in Linux

%after running the tar command, Linux will display the names of the files in the *flight.tgz* archive

[root@sandbox ~]# hadoop fs -put /root/airlines.dat /user/pig

[root@sandbox ~]# hadoop fs -put /root/routes.dat /user/pig

[root@sandbox ~]# hadoop fs -put /root/airports.dat /user/pig

[root@sandbox ~]# hadoop fs -ls /user/pig

[root@sandbox ~]# pig

1) (2 pts) List the Airline ID and name of all *airlines* where the name includes "Air Canada". You search should be non-case sensitive and include "Air Canada" with or without the spaces.

airlines = LOAD '/user/pig/airlines.dat' USING PigStorage(',') AS (airlineid:chararray, name:chararray, alias:chararray, iata:chararray, icao:chararray, callsign:chararray, country:chararray, active:chararray);

idcanada= FOREACH *airlines* GENERATE airlineid, LOWER(name) as name ;

matchcanada= FILTER idcanada BY name MATCHES '.*air(.)canada.*'; %The * matches 0 or more times. (.) matches any character in between. The two . in the beginning and end matches characters air canada

DUMP matchcanada;

```
(330,air canada)
(983,air canada jazz)
(2442,fortunair canada)
(19675,rainbow air canada)
```

2) (2 pts) Find the number of *airports* in each country. Submit the first five countries with the highest number of *airports*, together with the country names.

airports = LOAD '/user/pig/airports.dat' USING PigStorage(',') AS (airportid:chararray, name:chararray, city:chararray, country:chararray, iata:chararray, icao:chararray, lat:float,

lon:float, alt:float, timezone:chararray, dst:chararray, tztime:chararray, type:chararray, source:chararray);

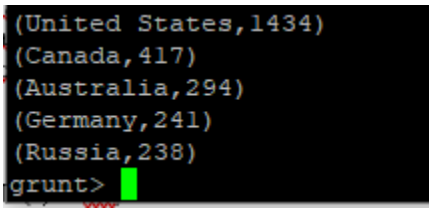
b_airports = GROUP *airports* BY country;

c_airports= FOREACH b_airports GENERATE GROUP AS country, COUNT(airports) AS cnt;

d_airports = ORDER c_airports BY cnt DESC;

e_airports = LIMIT d_airports 5;

DUMP e_airports;



3) (4 pts) Find the distinct *routes* between airports, based on source and destination airports. Submit the first five rows.

routes = LOAD '/user/pig/routes.dat' USING PigStorage(',') AS (airline: chararray, airlineid:int, sourceair:chararray, sourceairid:chararray, destair: chararray, destairid:chararray, codeshare: chararray, stops: int, equipment: chararray);

b_routes = FOREACH *routes* GENERATE sourceair, sourceairid, destair, destairid;

c_routes = DISTINCT b_routes;

d_routes = LIMIT c_routes 5;

DUMP d_routes;

4) (7 pts) Generate a table with source airport ID, source airport name, destination airport id, destination airport name and distance in kilometres using the output from the previous question.
Save your output in a tab separated file in an HDFS directory named *'routes with distances'*. Submit the screenshot of the directory listing and the first five lines of your output file.

Remember that you will have to get the latitude and longitude of each airport, using two joins-one for source and one for destination airport.

Each degree of latitude and longitude (close to the equator) is roughly 111 km. Calculate the distance in kilometres using the simple Euclidian formula:

distance = SQRT((lat2 - lat1) * (lat2 - lat1) + (lon2 - lon1) * (lon2 - lon1)) * 111

Haversine distance formula gives more accurate results when calculating the distance between two geographic coordinates. For the purposes of this exercise, however, the Euclidian distance would be sufficient.

routes = LOAD '/user/pig/routes.dat' USING PigStorage(',') AS (airline:chararray, airlineid:int, sourceair:chararray,sourceairid:chararray,destair: chararray, destairid: chararray,codeshare:chararray,stops: int, equipment:chararray);

airports = load '/user/pig/airports.dat' USING PigStorage(',') AS (airportid:chararray,name:chararray, city: chararray,country: chararray,ITA:chararray,ICAO:chararray,lat:float,lon:float,alt:float,timezone:chararray,dst:chararray,tztime:chararray,type:chararray,source:chararray);

b_routes = FOREACH *routes* GENERATE sourceair, sourceairid, destair, destairid;

c_routes= DISTINCT b_routes;

b_air= FOREACH *airports* GENERATE airportid, name, lat, lon;

c_air= DISTINCT b_air;

routeair= JOIN c_routes by sourceairid, c_air by airportid; %The **JOIN** operator is used to combine records from two or more relations. While performing a join operation, we declare one (or a group of) tuple(s) from each relation, as keys. When these keys match, the two particular tuples are matched, else the records are dropped.

b = FOREACH routeair GENERATE c_routes::sourceairid AS startid, c_routes::sourceair

AS startairport, c_air::name AS startname, c_routes::destair AS endairport,

c_routes::destairid AS endid, c_air::lat as lat1, c_air::lon as lon1;

```
c = JOIN b BY endid, c_air BY airportid;  %After JOIN, COGROUP, CROSS, or FLATTEN operations, the field names have the orginial alias and the disambiguate operator ( :: ) prepended in the schema. The disambiguate operator is used to identify field names in case there is an ambiguity.

d = FOREACH c GENERATE b::startid, b::startname, b::endid, c_air::name AS endname, SQRT((double)(c_air::lat-b::lat1)*(double)(c_air::lat-b::lat1)+(double)(c_air::lon-b::lon1)*(double)(c_air::lon-b::lon1))*(double)111 AS distance; %need to assign (double) for each operation as IMPLICIT_CAST_TO_DOUBLE warning will appear

f= LIMIT d 10;

DUMP f;
```

```
(1,Goroka Airport,2,Madang Airport,106.615143469184 72)
(2,Madang Airport,1,Goroka Airport,106.615143469184 72)
(3,Mount Hagen Kagamuga Airport,1,Goroka Airport,124.9021179176325)
(4,Nadzab Airport,1,Goroka Airport,157.67349789443568)
(4,Nadzab Airport,2,Madang Airport,183.56825147649573)
(5,Port Moresby Jacksons International Airport,1,Goroka Airport,424.748108336311 93)
(5,Port Moresby Jacksons International Airport,2,Madang Airport,496.332648686155 15)
(6,Wewak International Airport,2,Madang Airport,296.3787117119872)
(5430,Momote Airport,2,Madang Airport,393.4699839198462)
(5436,Vanimo Airport,2,Madang Airport,570.8536271144582)
```

```
e = STORE d into '/user/pig/routes_with_distances' USING PigStorage(',');
```