**Analysis of Medicare Plans**

*Author: Prashanth Shantigrama*

*Owner: Prashanth Shantigrama***Document History**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Revision Number | Revision Date | Summary of Changes | Revised By |
| 1.0 | 6/30/2015 | Document Creation | Prashanth Shantigrama |
| 1.1 | 09/29/2017 | Work Done for IBM Certification in Hadoop | Srinivas Bikki |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

1. Introduction 4

1.1 Problem Statment 4

1.2 Motivation 4

1.3 Objectives 4

1.4 Analysis Required for this project 4

2. Solution Architecture 6

2.1 Architecture Components 6

2.2 System Requirements 7

2.3 Data Source 7

3. Project Work Flow 10

4. Analysis Performed 11

5. Code 12

5.1 Pig Latin Scripts 12

5.2 HIVE Scripts 13

6. Issues & Workarounds 25

# Introduction

## Problem Statment

Several Medicare plans are available for senior citizens and other qualified members to enrol into every year, that are offered by different health insurance companies offering these plans. While a lot of information regarding individual plans exist, it is difficult to compare plans based on various criteria to make an informed choice to suit unique situations of individual members as well as for plan benefit designers to compare plans and design benefits that meets unique requirements and are competitive in different markets. The purpose of this document is to detail the analysis of the medicare plans data across US and provide useful summary details.

## Motivation

* The primary purpose of this project is to facilitate analysis of Medicare plans to provide meaningful insights that help in choosing appropriate medicare plans by comparing all relevant details regarding these plans that are available in each counties throughout the country.
* While CMS provides rich details on all the plans that are offered county wise, it makes better analysis when each plans that are offered are compared on the finer descriptions of its cost and coverage details

## Objectives

* To implement an efficient system to extract , load and transform all data related to Medicare plans to perform analytics.
* Analysis of Medicare plans to compare plan offerings by various criteria’s to select suitable plan for the Members
* Analysis of Medicare plans to compare plan offerings to design suitable benefit plan for different regions

## Analysis Required for this project

1. Identify top 5 plans with lowest premiums for a given county across the  US
2. To find plans that have highest co-pays for doctors in a given county
3. To compare plans based on features like plans that offer free ambulance services
4. To compare plans based on  features like the benefits available for diabetes under specific plan
5. To compare plan benefits on diabetes and mental healthcare offered by all companies in a particular county

# Solution Architecture



## Architecture Components

The proposed solution uses Hadoop framework and its eco-system tools to implement a distributed storage and processing of all data to perform the analysis.

**Client Machine:** The medicare files will be downloaded from medicare internet location onto client machine.

**HDFS:** The downaloded files will be uploaded onto Hadoop Distributed Files System. The files are uploaded onto Cloudera VM ware which runs in Psudonode mode.

**PIG:** The raw medicare files on the HDFS will be cleansed and transformed on HDFS and spooled into CSV files on HDFS location using Pig Latin Scripts.

**HIVE:** The Partioned HIVE tables are built over the cleansed medicare plan info and plan services CSV files that are stored in HDFS which were generated by pig latin scripts.

**SQOOP:**Squoop can be used to export out the Analyzed summary data stored in HIVE tables to traditional RDBMS.

## System Requirements

**Softwares:** Pig Latin, HIVE, Sqoop and WINSCP

**Hardware:** Cloudera VM ware

**Cluster:** Psudo mode Cluster

**Libraries:**

## Data Source

The medicare data can be downloaded from medicare government website from the following location:

<https://www.medicare.gov/download/downloaddb.asp>

Specifically the following two CSV files consists of the plan information and plan services data from where the analysis of the data can be performed:

1. **PlanInfoCounty\_FipsCodeMoreThan30000**

This file contains the contractid, planid, segmented, planname and county ID where the county ID is more than 30000. The sample of the plan info table:

"H0022","001","0","2015","Buckeye Health Plan - MyCareOhio","Buckeye Health Plan - MyCareOhio (Medicare-Medicaid Plan)","","Cleveland Dayton Toledo and surrounding counties","1","For Profit","Con Fines de Lucro","48","Medicare-Medicaid Plan","http://mmp.bchpohio.com","","www.bchpohio.com","www.bchpohio.com","Approved by Medicare and Medicaid","Aprobadopor Medicare y Medicaid","FALSE","TRUE","FALSE","88","&nbsp;","","","<ul><li>This is a Medicare-Medicaid plan for people with both Medicare and Medicaid. Contact the plan for details.</li></ul>","","","","BUCKEYE COMMUNITY HEALTH PLAN INC.","","","","","4349 Easton Way Suite 200","Columbus","OH","43219","nslee@centene.com","1-866-549-8289","1-866-549-8289","711","711","nslee@centene.com","1-866-549-8289","1-866-549-8289","711","711","","4349 Easton Way Suite 200","Columbus","OH","43219","nslee@centene.com","1-866-549-8289","1-866-549-8289","711","711","nslee@centene.com","1-866-549-8289","1-866-549-8289","711","711","FALSE","FALSE","0","Not SNP","No hay planes para necesidades especiales","$0.00","$0.00","$0.00","$0.00","FALSE","39023"

"H0022","001","0","2015","Buckeye Health Plan - MyCareOhio","Buckeye Health Plan - MyCareOhio (Medicare-Medicaid Plan)","","Cleveland Dayton Toledo and surrounding counties","1","For Profit","Con Fines de Lucro","48","Medicare-Medicaid Plan","http://mmp.bchpohio.com","","www.bchpohio.com","www.bchpohio.com","Approved by Medicare and Medicaid","Aprobadopor Medicare y Medicaid","FALSE","TRUE","FALSE","88","&nbsp;","","","<ul><li>This is a Medicare-Medicaid plan for people with both Medicare and Medicaid. Contact the plan for details.</li></ul>","","","","BUCKEYE COMMUNITY HEALTH PLAN INC.","","","","","4349 Easton Way Suite 200","Columbus","OH","43219","nslee@centene.com","1-866-549-8289","1-866-549-8289","711","711","nslee@centene.com","1-866-549-8289","1-866-549-8289","711","711","","4349 Easton Way Suite 200","Columbus","OH","43219","nslee@centene.com","1-866-549-8289","1-866-549-8289","711","711","nslee@centene.com","1-866-549-8289","1-866-549-8289","711","711","FALSE","FALSE","0","Not SNP","No hay planes para necesidades especiales","$0.00","$0.00","$0.00","$0.00","FALSE","39035"

1. **PlanInfoCounty\_FipsCodeLessThan30000**

This file contains the contractid, planid, segmented, planname and county ID where the county ID is less than 30000. The sample data of the plan info table:

"H0028","001","0","2015","CHA HMO Inc.","Humana Gold Plus H0028-001 (HMO)","","Cedar Rapids Metro Area","1","For Profit","Con Fines de Lucro","1","HMO","www.humana-medicare.com","www.humana.com","https://www.humana.com/pharmacy/medicare/tools/druglist/","https://www.humana.com/pharmacy/medicare/","Approved by Medicare","Aprobadopor Medicare","FALSE","TRUE","FALSE","88","&nbsp;","","","<ul><li>This plan does not charge an annual deductible for all drugs. The $320 annual deductible only applies to drugs on certain tiers.</li></ul>","","","","CHA HMO INC.","","1501-2000 physicians and providers.","1501-2000 médicos y proveedores.","","500 West Main Street","Louisville","KY","40202","HumanaOCS@humana.com","1-800-833-2364","1-800-833-2364","711","711","","1-800-457-4708","1-800-457-4708","711","711","","500 West Main Street","Louisville","KY","40202","HumanaOCS@humana.com","1-800-833-2364","1-800-833-2364","711","711","","1-800-457-4708","1-800-457-4708","711","711","FALSE","FALSE","0","Not SNP","No hay planes para necesidades especiales","$0.00","$0.00","$0.00","$0.00","FALSE","19011"

"H0028","001","0","2015","CHA HMO Inc.","Humana Gold Plus H0028-001 (HMO)","","Cedar Rapids Metro Area","1","For Profit","Con Fines de Lucro","1","HMO","www.humana-medicare.com","www.humana.com","https://www.humana.com/pharmacy/medicare/tools/druglist/","https://www.humana.com/pharmacy/medicare/","Approved by Medicare","Aprobadopor Medicare","FALSE","TRUE","FALSE","88","&nbsp;","","","<ul><li>This plan does not charge an annual deductible for all drugs. The $320 annual deductible only applies to drugs on certain tiers.</li></ul>","","","","CHA HMO INC.","","1501-2000 physicians and providers.","1501-2000 médicos y proveedores.","","500 West Main Street","Louisville","KY","40202","HumanaOCS@humana.com","1-800-833-2364","1-800-833-2364","711","711","","1-800-457-4708","1-800-457-4708","711","711","","500 West Main Street","Louisville","KY","40202","HumanaOCS@humana.com","1-800-833-2364","1-800-833-2364","711","711","","1-800-457-4708","1-800-457-4708","711","711","FALSE","FALSE","0","Not SNP","No hay planes para necesidades especiales","$0.00","$0.00","$0.00","$0.00","FALSE","19019"

1. **vwPlanServices**

This file contains the contractid, category description, category code and benefit information. The sample data of the plan services table:

English,2015,H0001,001,0,Monthly Premium Deductible and Limits on How Much You Pay for Covered Services,1,In 2015 the monthly Part B Standard Premium is <b>$104.90</b>,Base Plan,000,1

Spanish,2015,H0001,001,0,Prima Mensual Deducible y Límites a la Cantidad que UstedPagaporlosServicios Cubiertos,1,La prima estándar que pagapor la Parte B en el 2015 es<b>$104.90</b>,Plan Básico,000,1

# Project Work Flow

**Source Files**

The source files were downloaded from the medicare government website on to the client machine and decompressed to CSV files.

**HDFS**

The files from the client machine are uploaded onto HDFS using the Hadoop FS commands. As the files are sample medicare files and uploaded onto psudo mode of cloudera VM ware all the files are stored on a single data node on HDFS.

**Clean up files using PIG Latin Scripts**

* The data in the plan info and plan services files had double quotes enclosed around each field. The files were cleaned using the pig scripts which will be listed in next section.
* The data in the plan services file had 2 rows for each record one in English and other in Spanish. The files was cleaned using a separate relation in the pig script.
* Plan Info & Plan services files were filtered with records where contractId, SegmentId, PlanId&CountyId were null using pig scripts. The details will be listed in next section.

**HIVE tables to analyze tables:**

Hive tables were built with the cleaned output files from the pig latin scripts. A custom JAVA code was written to order the filtered records into ascending or descenting order. The custom code was compiled into a JAR file and was deployed onto HDFS. The JAR files was added onto HIVE and the summary queries were developed.

**Export Data using Sqoop:**

The summary files data from HIVE tables can be exported out of HDFS into traditional databases using the Sqoop. However, this step is not in the scope of the project but can be included in the future use cases.

# Analysis Performed

The following are some of the analysis performed

* Grouping the plans based on companies managing the plans and the counties where offered.
* Finding plans that offer specific services like free ambulance service in its coverage descriptions.
* Comparing plans based on the benefits offered for specific conditions like Diabetes
* Comparing plans based on premiums and co-pays for specific coverage criteria like doctors co-pays
* Comparing plans based on premiums for specific coverage criteria like doctors co-pays

# Code

## Pig Latin Scripts

-- load up the base UDF (piggybank)

register /home/hadoop/medicare/piggybank.jar;

-- load the PlanInfoCounty\_FipsCodeLessThan30000 file and declare its structure

inputFileLT = LOAD 'medicare/PlanInfoCountyLT.csv' using

org.apache.pig.piggybank.storage.CSVExcelStorage(',', 'NO\_MULTILINE', 'UNIX', 'SKIP\_INPUT\_HEADER') as (contract\_id,plan\_id,segment\_id,contract\_year,org\_name,plan\_name,sp\_plan\_name,geo\_name,tax\_stus\_cd,

tax\_status\_desc,sp\_tax\_status\_desc,plan\_type,plan\_type\_desc,web\_address,partd\_wb\_adr,frmlry\_wbst\_adr,phrmcy\_wbst\_adr,fed\_approval\_status,sp\_fed\_approval\_status,

pos\_available\_flag,mail\_ordr\_avlblty,cvrg\_gap\_ofrd,cvrg\_gap\_ind,cvrg\_gap\_desc,contract\_important\_note,sp\_contract\_important\_note,plan\_important\_note,

sp\_plan\_important\_note,segment\_important\_note,sp\_segment\_important\_note,legal\_entity\_name,trade\_name,network\_english,network\_spanish,contact\_person,

street\_address,city,state\_code,zip\_code,email\_prospective,local\_phone\_prospective,tollfree\_phone\_prospective,local\_tty\_prospective,

tollfree\_tty\_prospective,email\_current,local\_phone\_current,tollfree\_phone\_current,local\_tty\_current,tollfree\_tty\_current,contact\_person\_pd,

street\_address\_pd,city\_pd,state\_code\_pd,zip\_code\_pd,email\_prospective\_pd,local\_phone\_prospective\_pd,tollfree\_phone\_prospective\_pd,local\_tty\_prospective\_pd,

tollfree\_tty\_prospective\_pd,email\_current\_pd,local\_phone\_current\_pd,tollfree\_phone\_current\_pd,local\_tty\_current\_pd,tollfree\_tty\_current\_pd,ma\_pd\_indicator,

ppo\_pd\_indicator,snp\_id,snp\_desc,sp\_snp\_desc,lis\_100,lis\_75,lis\_50,lis\_25,regional\_indicator,CountyFIPSCode);

-- load the PlanInfoCounty\_FipsCodeLessThan30000 file and declare its structure

inputFileMT = LOAD 'medicare/PlanInfoCountyMT.csv' using

org.apache.pig.piggybank.storage.CSVExcelStorage(',', 'NO\_MULTILINE', 'UNIX', 'SKIP\_INPUT\_HEADER') as (contract\_id,plan\_id,segment\_id,contract\_year,org\_name,plan\_name,sp\_plan\_name,geo\_name,tax\_stus\_cd,

tax\_status\_desc,sp\_tax\_status\_desc,plan\_type,plan\_type\_desc,web\_address,partd\_wb\_adr,frmlry\_wbst\_adr,phrmcy\_wbst\_adr,fed\_approval\_status,sp\_fed\_approval\_status,

pos\_available\_flag,mail\_ordr\_avlblty,cvrg\_gap\_ofrd,cvrg\_gap\_ind,cvrg\_gap\_desc,contract\_important\_note,sp\_contract\_important\_note,plan\_important\_note,

sp\_plan\_important\_note,segment\_important\_note,sp\_segment\_important\_note,legal\_entity\_name,trade\_name,network\_english,network\_spanish,contact\_person,

street\_address,city,state\_code,zip\_code,email\_prospective,local\_phone\_prospective,tollfree\_phone\_prospective,local\_tty\_prospective,

tollfree\_tty\_prospective,email\_current,local\_phone\_current,tollfree\_phone\_current,local\_tty\_current,tollfree\_tty\_current,contact\_person\_pd,

street\_address\_pd,city\_pd,state\_code\_pd,zip\_code\_pd,email\_prospective\_pd,local\_phone\_prospective\_pd,tollfree\_phone\_prospective\_pd,local\_tty\_prospective\_pd,

tollfree\_tty\_prospective\_pd,email\_current\_pd,local\_phone\_current\_pd,tollfree\_phone\_current\_pd,local\_tty\_current\_pd,tollfree\_tty\_current\_pd,ma\_pd\_indicator,

ppo\_pd\_indicator,snp\_id,snp\_desc,sp\_snp\_desc,lis\_100,lis\_75,lis\_50,lis\_25,regional\_indicator,CountyFIPSCode);

--Filter records

inputFileFlterLT= FILTER inputFileLT BY ((contract\_id is not null) and (plan\_id is not null) and (segment\_id is not null) and (CountyFIPSCode is not null));

inputFileFlterMT= FILTER inputFileMT BY ((contract\_id is not null) and (plan\_id is not null) and (segment\_id is not null) and (CountyFIPSCode is not null));

-- loop thru the input data to clean up the required fields a bit

cleanedUpPlanLT = foreach inputFileFlterLT GENERATE contract\_id, plan\_id, segment\_id, contract\_year, org\_name, plan\_name, CountyFIPSCode;

cleanedUpPlanMT = foreach inputFileFlterMT GENERATE contract\_id, plan\_id, segment\_id, contract\_year, org\_name, plan\_name, CountyFIPSCode;

-- Combine the 2 Plans into 1 table

combinedPlanInfo = UNION cleanedUpPlanLT, cleanedUpPlanMT;

-- save results back to HDFS STORE

store combinedPlanInfo into 'medicare/output/PlanInfoCounty.csv' USING PigStorage(',');

-- load the vwPlanServices file and declare its structure

inputPlanServices = LOAD 'medicare/vwPlanServices.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',', 'NO\_MULTILINE', 'UNIX', 'SKIP\_INPUT\_HEADER')

as (language:chararray,contract\_year:chararray,contract\_id:chararray, plan\_id:chararray,

segment\_id:chararray,categoryDescription:chararray,

categoryCode:long,benefit:chararray,

package\_name:chararray,package\_id:chararray,sentences\_sort\_order:int);

-- trim down to just 'English'

englishPlanServices = FILTER inputPlanServices BY ((language == 'English') and (contract\_id is not null) and (plan\_id is not null) and (segment\_id is not null));

-- loop thru the input data to clean up the required fields a bit

cleanedPlanServices = foreach englishPlanServices GENERATE contract\_id, plan\_id, segment\_id, contract\_year, categoryDescription, categoryCode, benefit;

-- save results back to HDFS STORE

store cleanedPlanServices into 'medicare/output/PlanServices.csv' USING PigStorage(',');

## Hive Scripts

-- Load Pig data PlanInfoCounty into hive table

CREATE EXTERNAL TABLE PlanInfoCounty (contract\_id STRING,

plan\_id STRING,

segment\_id STRING,

contract\_year STRING,

org\_name STRING,

plan\_name STRING,

CountyFIPSCode STRING

)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

STORED AS TEXTFILE

LOCATION '/user/hadoop/medicare/output/PlanInfoCounty.csv';

-- Load Pig data PlanInfoCounty into hive table

CREATE EXTERNAL TABLE PlanServices (contract\_id STRING,

plan\_id STRING,

segment\_id STRING,

contract\_year STRING,

categoryDescription STRING,

categoryCode STRING,

benefit STRING

)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

STORED AS TEXTFILE

LOCATION '/user/hadoop/medicare/output/PlanServices.csv';

**Hive Tables/Queries**:

1. **Identify top 5 plans with lowest premiums for a given county across the US**

CREATE TABLE PREMIUM\_PLANS AS

SELECT

pic.contract\_id,

pic.plan\_id,

pic.plan\_name,

cast(pic.CountyFIPSCode as BIGINT) as county,

ps.categoryDescription,

cast(regexp\_extract(ps.benefit,'<[a-z]>[$](.\*)</[a-z]>', 1) as FLOAT) as lowPremium

FROM PlanInfoCounty pic

JOIN PlanServices ps

ON (pic.contract\_id = ps.contract\_id

AND pic.plan\_id = ps.plan\_id)

WHERE ps.benefit LIKE '%remium%'

AND ps.categoryCode="1"

AND regexp\_extract(ps.benefit,'<[a-z]>[$](.\*)</[a-z]>', 1)> 0

AND CountyFIPSCode IS NOT NULL

ORDER BY county, lowPremium ASC;

--Rank it by county will generate all 5 lowPremium Plans

CREATE TABLE LOWPREMIUM\_PLANS AS

SELECT

contract\_id,

plan\_id,

plan\_name,

county,

lowPremium

FROM(

SELECT inn.\*,row\_number() over (PARTITION BY inn.county ORDER BY inn.lowPremium ASC) as rank

FROM

(SELECT distinct contract\_id,plan\_id, plan\_name,county,lowPremium FROM PREMIUM\_PLANS) inn

) ranked\_table

WHERE ranked\_table.rank <= 5

AND county IS NOT NULL AND lowPremium IS NOT NULL

ORDER BY county,lowPremium ASC;

SELECT \* FROM LOWPREMIUM\_PLANS WHERE county=72009;

1. **To find plans that have highest co-pays for doctors in a given county**

CREATE TABLE COPAY\_PLANS AS

SELECT

pic.contract\_id,

pic.plan\_id,

pic.plan\_name,

cast(pic.CountyFIPSCode as BIGINT) as county,

ps.categoryDescription,

cast(regexp\_extract(ps.benefit,'<[a-z]>[$](.\*)</[a-z]>', 1) as FLOAT) as copay

FROM PlanInfoCounty pic

JOIN PlanServices ps

ON (pic.contract\_ID = ps.Contract\_ID

AND pic.plan\_id = ps.Plan\_ID)

WHERE ps.categoryDescription LIKE '%Doctor%office%visits%'

AND regexp\_extract(ps.benefit,'<[a-z]>[$](.\*)</[a-z]>', 1)> 0

AND CountyFIPSCode IS NOT NULL

ORDER BY county, copay DESC;

CREATE TABLE HIGHCOPAY\_PLANS AS

SELECT

contract\_id,

plan\_id,

plan\_name,

county,

copay

FROM(SELECT inn.\*,

row\_number() over (PARTITION BY inn.county ORDER BY inn.copay DESC) as row\_num

FROM (SELECT distinct contract\_id,plan\_id, plan\_name,county,copay FROM COPAY\_PLANS) inn

) rn

WHERE rn.row\_num <= 5

AND county IS NOT NULL AND copay IS NOT NULL

ORDER BY county,copay DESC;

SELECT \* FROM HIGHCOPAY\_PLANS WHERE county=72009;

1. **To compare plans based on features like plans that offer free ambulance services**

CREATE TABLE AMB\_PLANS AS

SELECT

pic.contract\_id,

pic.plan\_id,

pic.plan\_name,

cast(pic.CountyFIPSCode as BIGINT) as county,

ps.categoryCode,

ps.categoryDescription,

substr(benefit,(instr(benefit,'<b>')+3), instr(benefit,'</b>')-(instr(benefit,'<b>')+3)) as amb\_service

FROM PlanInfoCounty pic

JOIN PlanServices ps

ON (pic.contract\_id = ps.contract\_id

AND pic.plan\_id = ps.plan\_id)

WHERE ps.categoryCode="5"

AND (ps.benefit rlike '.\*You pay nothing.\*' or ps.benefit rlike '.\*In-network.\*' or ps.benefit rlike '.\*Out-of-network.\*')

AND CountyFIPSCode IS NOT NULL;

SELECT \* FROM AMB\_PLANS WHERE county=72009;

SELECT \* FROM AMB\_PLANS LIMIT 50;

1. **To compare plans based on features like the benefits available for diabetes under specific plan**

SELECT

DISTINCT

pic.contract\_id,

pic.plan\_id,

pic.plan\_name,

cast(pic.CountyFIPSCode as BIGINT) as county,

ps.categoryCode,

ps.categoryDescription,

ps.benefit

FROM PlanInfoCounty pic

JOIN PlanServices ps

ON (pic.contract\_id = ps.contract\_id

AND pic.plan\_id = ps.plan\_id)

WHERE ps.categoryCode="8"

OR (ps.categoryCode!="8" AND ps.benefit like '%diabetes%')

AND CountyFIPSCode IS NOT NULL;

1. **To compare plan benefits on diabetes and mental healthcare offered by all companies in a**

**particular county**

SELECT

DISTINCT

pic.contract\_id,

pic.plan\_id,

pic.plan\_name,

cast(pic.CountyFIPSCode as BIGINT) as county,

ps.categoryCode,

ps.categoryDescription,

ps.benefit

FROM PlanInfoCounty pic

JOIN PlanServices ps

ON (pic.contract\_id = ps.contract\_id

AND pic.plan\_id = ps.plan\_id)

WHERE

ps.categoryCode="8" OR (ps.categoryCode="16")

limit 10;

## Twitter Analysis (FLUME / HIVE)

**Step 1: Create and application in Twitter**

https://apps.twitter.com

**Step 2: Note the Consumer Key,Consumer Secret,Access Token & Access Token Secret**

**from the application created**

**Step 3: Download the jar file flume-sources-1.0-SNAPSHOT.jar and copy it into /usr/local/flume/lib**

This jar file is the source component for creating a Twitter Agent

( This is a flume configuration for connecting to Twitter Server and sinking the data into HDFS)

**Step 4. Build the Flume Agent configuration file.**

Here the Source is extracting the data from Twitter

Sink is writing data to HDFS

# describe and configure the source

TwitterAgent.sources = Twitter

TwitterAgent.channels = MemChannel

TwitterAgent.sinks = HDFS

TwitterAgent.sources.Twitter.type = com.cloudera.flume.source.TwitterSource

TwitterAgent.sources.Twitter.channels = MemChannel

TwitterAgent.sources.Twitter.consumerKey = dyxT0U8SLYXFkDJvCXSD5RHCz

TwitterAgent.sources.Twitter.consumerSecret = ErtB1AbJhsXhfJH7zYMA2z8nHsdUAjfT1Z4wz2TN6AsNwSuBqg

TwitterAgent.sources.Twitter.accessToken = 2788097081-MSrVLTejFhbHRN2AVtQ63LLiRFT5FymXkl7N4LL

TwitterAgent.sources.Twitter.accessTokenSecret = n2hpnOoc6TcmQ2EowKV924DhstuY5ZtTEQ6TPGOFSfL3U

# Twitter handles to search

TwitterAgent.sources.Twitter.keywords = AARP,Aetna,BlueCross,Humana,UnitedHealthcare

# describe and configure the sink

TwitterAgent.sinks.HDFS.channel = MemChannel

TwitterAgent.sinks.HDFS.type = hdfs

TwitterAgent.sinks.HDFS.hdfs.path = /user/hadoop/twitter/flume\_data

TwitterAgent.sinks.HDFS.hdfs.fileType = DataStream

TwitterAgent.sinks.HDFS.hdfs.writeFormat = Text

TwitterAgent.sinks.HDFS.hdfs.batchSize = 1000

TwitterAgent.sinks.HDFS.hdfs.rollSize = 0

TwitterAgent.sinks.HDFS.hdfs.rollCount = 10000

TwitterAgent.sinks.HDFS.hdfs.rollInterval = 600

# describe and configure the channel

TwitterAgent.channels.MemChannel.type = memory

TwitterAgent.channels.MemChannel.capacity = 1000

TwitterAgent.channels.MemChannel.transactionCapacity = 10

**5. Run the Flume Agent to collect the data from Twitter**

flume-ng agent -n TwitterAgent -f TwitterAgent.conf

**Hive: Create table and Perform analysis**

CREATE EXTERNAL TABLE tweets (

id BIGINT,

created\_at STRING,

source STRING,

favorited BOOLEAN,

retweeted\_status STRUCT<

text:STRING,

user:STRUCT<screen\_name:STRING,name:STRING>,

retweet\_count:INT>,

entities STRUCT<

urls:ARRAY<STRUCT<expanded\_url:STRING>>,

user\_mentions:ARRAY<STRUCT<screen\_name:STRING,name:STRING>>,

hashtags:ARRAY<STRUCT<text:STRING>>>,

text STRING,

user STRUCT<

screen\_name:STRING,

name:STRING,

friends\_count:INT,

followers\_count:INT,

statuses\_count:INT,

verified:BOOLEAN,

utc\_offset:INT,

time\_zone:STRING>,

in\_reply\_to\_screen\_name STRING

)

ROW FORMAT SERDE 'com.cloudera.hive.serde.JSONSerDe';

LOCATION '/user/cloudera/twitter'

select t.retweeted\_screen\_name, sum(retweets) as total\_retweets, count(\*) as tweet\_count

from

(

select retweeted\_status.user.screen\_name as retweeted\_screen\_name, retweeted\_status.text, max(retweeted\_status.retweet\_count) as retweets from tweets group by retweeted\_status.user.screen\_name,retweeted\_status.text

)

t

group by t.retweeted\_screen\_name

order by total\_retweets DESC, tweet\_count ASC

limit 10;

# Issues & Workarounds