Medicare Plans Analytics

Author: Rama Tripathy

Table of Contents

Introduction

1.1 Problem Statement

Several Medicare plans are available for senior citizens and other qualified members to enroll into every year that are offered by different health insurance organizations. 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 comparison and summary details.

1.2 Motivation

- The primary purpose of this project is to facilitate analysis of Medicare plans using Hadoop to provide meaningful insights.
- Help in choosing appropriate Medicare plan by comparing all relevant details for various 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 plan that are offered are compared on the finer descriptions of its cost and coverage details

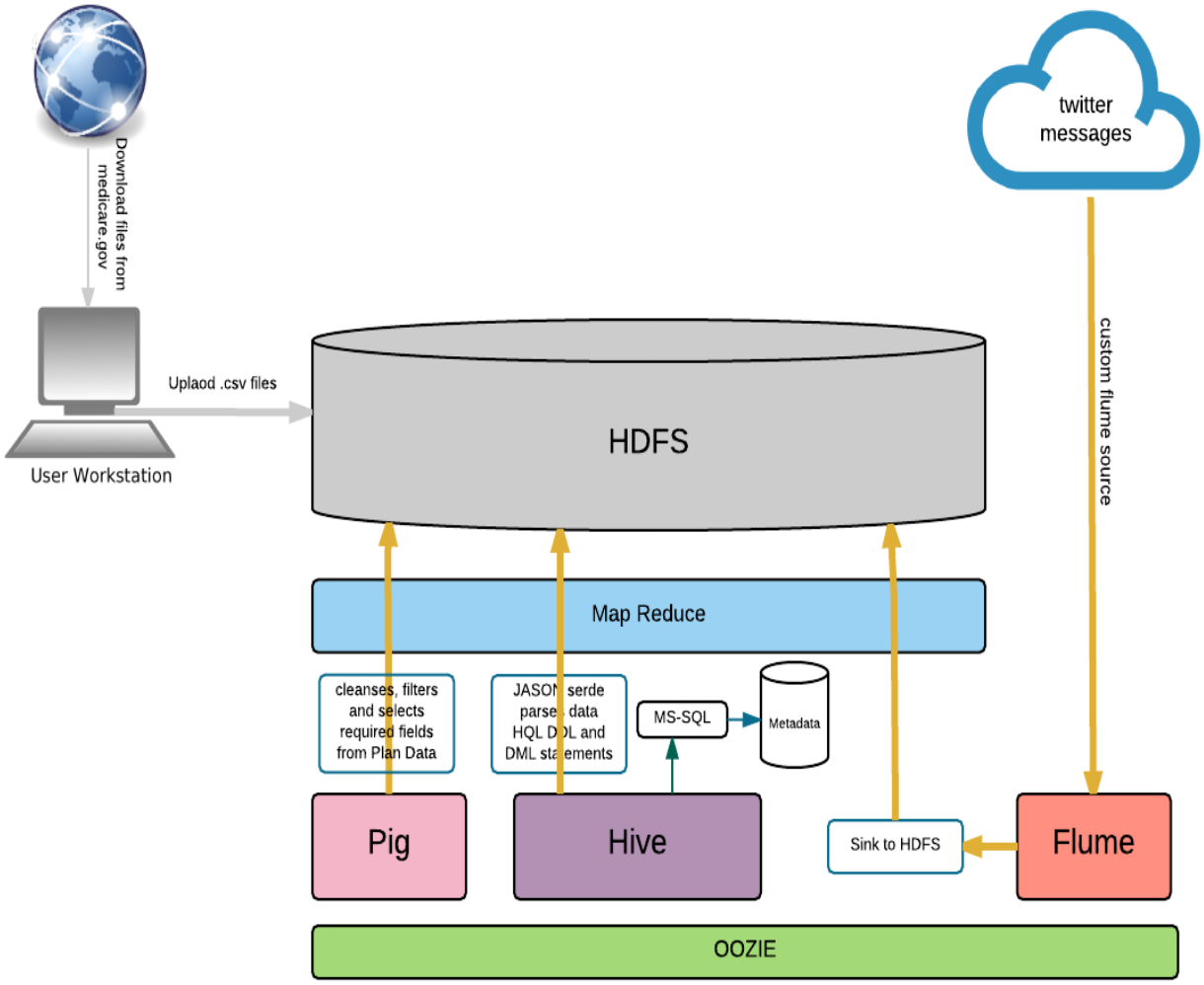
1.3 Objectives

- 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 Medicare plan providers to select suitable plan for the Members
- Analysis of Medicare plans to compare plan offerings to design suitable benefit plan for different regions
- Member sentiment analysis using twitter feeds

1.4 Analysis required for this project

1. Identify top 5 plans with lowest premiums with the current and previous star ratings for a given county across the US
2. To find plans that have lowest 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
6. To find out plan providers having maximum twitter presence

Solution Architecture



1.5 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 provided by CMS onto client machine.

Twitter Source: The twitter messages containing information for organizations are collected using **Flume** tool with twitter application as a custom source and sinked into HDFS.

HDFS: The downloaded files will be uploaded onto Hadoop Distributed Files System. The files are uploaded onto Cloudera VMware which runs in Psudonode mode.

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

HIVE: The HIVE tables with bucket cluster are built over the cleansed Medicare plan text files generated by pig latin scripts. JASON serde jar function is used by Hive queries to process the twitter messages and converted to tables and the text field in these table rows are analyzed using HQL queries.

1.6 System Requirements

Softwares and Tools: Cloudera VMware 11 and above, Pig Latin, HIVE, Flume, WINSCP, Ubuntu, JDK 1.7, MySQL, SuperPutty, Hadoop 2.6.0

Hardware: Cloudera VM ware

Cluster: Pseudo mode Cluster

Libraries: Piggybank, MS-SQL jar, serde jar

1.7 Data Sources

The Medicare data can be downloaded from medicare.gov website from the following location: https://www.medicare.gov/download/downloaddb.asp

For our analysis, we need the following five csv files (Appendix-A describe the fields for each of these five files)

- ***PlanInfoCounty_FipsCodeMoreThan30000*** **and** ***PlanInfoCounty_FipsCodeLessThan30000 :***

  These files are used to search plan information by county.

- ***vwPlanServices*****:** This file contains 11 fields and has the benefit information each plan provide:
- ***vwGeography:*** This information is used to search all plan information by geographical location.
- ***vwStarRating_SummaryScores*****:** This file provides summary level quality and performance data for plans:

Project Work Flow

1.8 Source Files

The source files were downloaded from the Medicare government website on to the client machine as CSV files.

1.9 HDFS

The files from the client machine are uploaded onto user working directory /home/hadoop on VMware workstation using WinSCP tool. Then using the following Hadoop FS commands, the csv files are loaded into hdfs. As the files are uploaded onto pseudo mode of cloudera VMware all the files are stored on a single data node on HDFS.

```
hdfs dfs -mkdir /medicareplan

hdfs dfs -put PlanInfoCounty_FipsCodeLessThan30000.csv /medicareplan/Plan1.csv

hdfs dfs -put PlanInfoCounty_FipsCodeMoreThan30000.csv /medicareplan/Plan2.csv

hdfs dfs -put vwGeography.csv /medicareplan/vwgeography.csv

hdfs dfs -put vwPlanServices.csv /medicareplan/vwplanservices.csv

hdfs dfs -put  vwStarRating_SummaryScores.csv  /medicareplan/vwstarsummary.csv
```

1.10 Clean up files using PIG Latin Scripts

- The data in the 5 csv files have double quotes enclosed around each field. The files were cleaned using the pig scripts and piggybank user defined function (udf) CSVExcelStorage().
- The data in the plan services file and Star Rating file had 2 rows for each record one in English and other in Spanish. The files was cleaned using a filter command in the pig script.
- All five csv files were filtered with records where the field were null using pig scripts and also having column headings.
- The pig scripts are explained in the Appendix-B: PIG LATIN SCRIPTS section.

1.11 Analyze the Data using Hive

Hive tables were built with the cleaned output files from the pig latin scripts. A custom JAVA code (hive udf) was written for the following functions:

a. Premium calculation
b. Copay calculation
c. Coins calculation

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 using HQL were developed.

- The HQL scripts are explained in the Appendix-C: Hive SCRIPTS section.

1.12 Twitter Feeds Imported into HDFS Using Flume and analyzed using Hive:

Flume is used to collect twitter data using a custom Twitter Source calss "flume-sources-1.0-SNAPSHOT.jar" and then the data is moved into HDFS. These hdfs files are then converted to hive structured tables using Jason serde class "hive-serdes-1.0-SNAPSHOT.jar". The flume configuration file "flume-conf.conf" file is configured with Twitter as the source, memory as the channel and HDFS as the sink and the twitter key words to include the Plan Provider names to capture twitter feed containing these keywords.

TwitterAgent.sources.Twitter.keywords=
Berkshire,Westmoreland,WellCare,VillageCareMAX,UCare's,SummaCare,StayWell,SelectCare,PrimeWest,PacificSource,OneCare,MedStar,Medicare,KelseyCare,InovaCares,IliniCare,IEHP,HUMANA,Humana,HealthSun,HealthSpring,HealthPlus,Healthfirst,GLOBALHEALTH,CDPHP,CareSource,CarePlus,CarePartners,CareOregon,CareMore,Care1st,BlueCross,BlueChoice,BlueCare,ArchCare,AmeriHealth,Amerigroup Healthcare,AltaMed,AlphaCare,AllCare,Aetna,Advicare

Member sentiment on the plan providers are analyzed using the hive structured table and the Plan Provider keywords.

The complete flume conf file and Hive queries are explained in **Appendix-D: Flume Setups.**

Analysis Performed

The following are some of the analysis performed:

1.13 Monthly Premium Calculation:

Some of the issues and challenges we have identified for premium calculation are:

- ✓ Some plans ask for monthly premium and others are for yearly
- ✓ There are plans with additional premium along with primary premium
- ✓ There are plans with Plan B premium along with primary premium and also some with both additional premium and Plan B premium

In 2015 the monthly Part B Standard Premium is $104.90

Examples of Benefit statement:

==========

"$29.45 monthly plan premium"

"Additional $21.20 per month. You must keep paying your Medicare Part B premium and your $20 monthly plan premium.

"$360 per year."

==========

1.14 Hive query optimization:

We analyzed and applied several optimization techniques and recommended the suitable techniques for this analysis.

Query used for comparison: Identify top 5 plans with lowest premiums for a given county

System configuration used:

- Windows system with 6 GB processor
- 4GB processor
- 32 GB disk space
- UBUNTU 14.4
- VMWare 12.0
- Hadoop 2.6.0
- Apache Hive 1.2.1
- Pig 0.15.0
- Apache flume 1.6.0

Tables used:

1. TABLE starratetable (ContractId STRING, summary_score STRING,star_rating_Current STRING, star_rating_previous STRING) : Total rows:
2. TABLE countytable (contract_id STRING, plan_id STRING, segment_id STRING, org_name STRING, plan_name STRING, plan_type STRING, countyFIPcode STRING)
3. TABLE planservicetable (contractid STRING, planid STRING, segmentid STRING,category STRING, categoryCode STRING , benefit STRING)
4. TABLE countynamestable (FIPScode STRING, countyname STRING)

1.14.1 Use of subqueries/temporary tables:

We can reduce the number of rows to be processed before the join operation by using subqueries/or temporarily storing the results in temporary tables. Surprisingly, it took more time by using these techniques as the number of mapreduce jobs resulted in using these was higher than the single query and resulted in higher processing times.

Hive QL with temporary table/subqueries:

```
SELECT contract_id,countyname,org_name, plan_name, max(premcalc(benefit)) prem,star_rating_Current, star_rating_previous
FROM countytable
JOIN (Select FIPScode,countyname from countynamestable where FIPScode = "25003" Limit 1) as cn
ON (countytable.countyFIPcode= cn.FIPScode)
JOIN (select planid,contractid,benefit from planservicetable
WHERE CategoryCode="1" AND (instr(benefit,"monthly premium")>0 OR instr(benefit,"per month")>0  OR instr(benefit,"per year")>0)) as ps
on (ps.planid = countytable.plan_id and ps.contractid =countytable.contract_id)
LEFT OUTER JOIN(select ContractId, star_rating_Current, star_rating_previous
FROM starratetable WHERE summary_score = 'Overall Star Rating') as sr
ON (sr.ContractId = countytable.contract_id)
GROUP BY contract_id,countyname,org_name, plan_name,star_rating_Current, star_rating_previous
order by prem  LIMIT 5;
```

```
Total Jobs=9

Time taken: 238.799 seconds, Fetched: 5 row(s)
```

QL: Without subqueries/temporary tables

```
SELECT contract_id,countyname,org_name, plan_name, max(premcalc(benefit)) prem,star_rating_Current, star_rating_previous
FROM countytable
join countynamestable ON (countynamestable.FIPScode = countytable.countyFIPcode)
JOIN planservicetable on (planservicetable.planid = countytable.plan_id and countytable.contract_id  = planservicetable.contractid)
LEFT OUTER JOINstarratetable ON (countytable.contract_id  = starratetable.ContractId)
WHERE countytable.countyFIPcode = "25003"
AND CategoryCode="1" AND (instr(benefit,"monthly premium")>0 OR instr(benefit,"per month")>0  OR instr(benefit,"per year")>0)
AND summary_score = 'Overall Star Rating'
GROUP BY contract_id,countyname,org_name, plan_name,star_rating_Current, star_rating_previous
order by prem LIMIT 5;
```

```
Total jobs = 6

Time taken: 151.862 seconds, Fetched: 5 row(s)
```

1.14.2 Partitioning Hive Tables:

The countytable is the largest table having 1180458 rows can be partitioned using Stae/CountyFIPSCode/Year. Using State has no relevance as the queries are on CountyFIPScode. Using CountyFIPSCode will result in too much fragmentation of data into HDFS blocks containing few rows, incurring serious overheads. The current analysis is only for 2015 year data, there is no use in partitioning using year. If multi-year data need to be analyzed, the hive table for countytable can be partitioned using year. For our analysis, we are not using this Partition technique.

1.14.3 Bucketing Hive Tables:

Since the queries are using countyFIPscode and this column is in the largest table countytable, we will use this optimization technique to bucket the tables using county code and compare the query processing results. We have defined the countybuckettable using countyFIPcode for bucket definition and use this table in our queries instead of countytable:

```
TABLE countybuckettable ( contract_id  STRING, plan_id  STRING,  segment_id  STRING,
org_name  STRING,  plan_name STRING,  plan_type STRING, countyFIPcode STRING)
CLUSTERED BY (countyFIPcode) INTO 64 BUCKETS;
```

QL: With subqueries/temporary tables using Bucket Tables

```
SELECT contract_id,countyname,org_name, plan_name, max(premcalc(benefit)) prem,star_rating_Current, star_rating_previous
FROM countybuckettable
JOIN (Select FIPScode,countyname from countynamestable WHERE FIPScode = "25003" Limit 1) as cn
ON (countybuckettable.countyFIPcode= cn.FIPScode)
JOIN (select planid,contractid,benefit from planservicetable
WHERE CategoryCode="1" AND (instr(benefit,"monthly premium")>0 OR instr(benefit,"per month")>0  OR instr(benefit,"per year")>0))
as ps
on (ps.planid = countybuckettable.plan_id and ps.contractid =countybuckettable.contract_id)
LEFT OUTER JOIN(select ContractId, star_rating_Current, star_rating_previous
from starratetable WHERE summary_score = 'Overall Star Rating') as sr
ON (sr.ContractId = countybuckettable.contract_id)
GROUP BY contract_id,countyname,org_name, plan_name,star_rating_Current, star_rating_previous
ORDER BY prem  LIMIT 5;
```

Total jobs = 9

Time taken: 192.863 seconds, Fetched: 5 row(s)

QL: With subqueries/temporary tables using Bucket Tables

```
SELECT contract_id,countyname,org_name, plan_name, max(premcalc(benefit)) prem,star_rating_Current, star_rating_previous
FROM countybuckettable
JOIN countynamestable ON (countynamestable.FIPScode = countybuckettable.countyFIPcode)
JOIN planservicetable on (planservicetable.planid = countybuckettable.plan_id and countybuckettable.contract_id  =
planservicetable.contractid)
LEFT OUTER JOINstarratetable ON (countybuckettable.contract_id  = starratetable.ContractId)
WHERE countybuckettable.countyFIPcode = "25003"
AND CategoryCode="1" AND (instr(benefit,"monthly premium")>0 OR instr(benefit,"per month")>0  OR instr(benefit,"per year")>0)
AND summary_score = 'Overall Star Rating'
GROUP BY contract_id,countyname,org_name, plan_name,star_rating_Current, star_rating_previous
ORDER BY prem  LIMIT 5;
```

Total jobs = 6

Time taken: 147.226 seconds, Fetched: 5 row(s)

Perform Analysis Summary Table:

	With subqueries/temporary tables	Without subqueries/ temporary tables
Without Bucket clustering	**Total Jobs: 9** **Time Taken: 266.399** **(Refer to Appendix-E: Screenshot1)**	**Total Jobs: 6** **Time Taken: 147.379** **(Refer to Appendix-E: Screenshot2)**
With Bucket clustering	**Total Jobs: 9** **Time Taken: 175.28** **(Refer to Appendix-E: Screenshot3)**	**Total Jobs: 6** **Time Taken: 145.938** **(Refer to Appendix-E: Screenshot4)**

1.15 Plan Data Analysis:

The following queries were run for CountyFIPScode= "25003". Refer to Appendix-C for the Hive query details:

- Grouping the plans based on companies managing the plans and the counties where offered.

Query Output:

H8578 BERKSHIRE HNE Medicare Advantage Plans HNE Medicare Basic No Rx (HMO) 123.9 4 out of 5 stars 4.5 out of 5 stars

H8578 BERKSHIRE HNE Medicare Advantage Plans HNE Medicare Value (HMO) 124.9 4 out of 5 stars 4.5 out of 5 stars

R7444 BERKSHIRE UnitedHealth care AARP Medicare Complete Choice (Regional PPO) 144.9 4 out of 5 stars 3.5 out of 5 stars

H8578 BERKSHIRE HNE Medicare Advantage Plans HNE Medicare Basic (HMO) 179.9 out of 5 stars 4.5 out of 5 stars

H8578 BERKSHIRE HNE Medicare Advantage Plans HNE Medicare Premium No Rx (HMO) 193.9 4 out of 5 stars 4.5 out of 5 stars

- Finding plans that offer specific services like free ambulance service in its coverage descriptions.

Query Output:

25003 Tufts Health Plan - Network Health Tufts Health Unify (Medicare-Medicaid Plan) 0.0

25003 HNE Medicare Advantage Plans HNE Medicare Premium No Rx (HMO) 20.0

25003 HNE Medicare Advantage Plans HNE Medicare Premium (HMO) 20.0

25003 HNE Medicare Advantage Plans HNE Medicare Plus (HMO) 30.0

25003 HNE Medicare Advantage Plans HNE Medicare Basic No Rx (HMO) 40.0

- Comparing plans based on the benefits offered for specific conditions like Diabetes

Query Output:

25003 UnitedHealthcare AARP MedicareComplete Choice (Regional PPO)

25003 Tufts Health Plan - Network Health Tufts Health Unify (Medicare-Medicaid Plan)

25003 HNE Medicare Advantage Plans HNE Medicare Value (HMO)

25003 HNE Medicare Advantage Plans HNE Medicare Premium No Rx (HMO)

25003 HNE Medicare Advantage Plans HNE Medicare Premium (HMO)

- Comparing plans based on premiums and co-pays for specific coverage criteria like doctors co-pays

Query Output:

25003 UnitedHealthcare AARP MedicareComplete Choice (Regional PPO)

25003 Tufts Health Plan - Network Health Tufts Health Unify (Medicare-Medicaid Plan)

25003 HNE Medicare Advantage Plans HNE Medicare Value (HMO)

25003 HNE Medicare Advantage Plans HNE Medicare Premium No Rx (HMO)

25003 HNE Medicare Advantage Plans HNE Medicare Premium (HMO)

- Comparing plans based on premiums for specific coverage criteria like doctors co-pays

Query Output:

25003 UnitedHealthcare AARP MedicareComplete Choice (Regional PPO)

25003 Tufts Health Plan - Network Health Tufts Health Unify (Medicare-Medicaid Plan)

25003 HNE Medicare Advantage Plans HNE Medicare Value (HMO)

25003 HNE Medicare Advantage Plans HNE Medicare Premium No Rx (HMO)

25003 HNE Medicare Advantage Plans HNE Medicare Premium (HMO)

- To find out plan providers having maximum twitter presence

Refer to Appendix-E: Screen shot-5 for the query output

Issues & Workarounds

- The data from CSV files were not consistent and was reading NULL values for certain fields. As a work around used the CSVExcelStorage() method from the Piggy bank to overcome this issue.
- Hive UDFs when copied to user home directory and called from hive queries, 'java.lang.IllegalAccessException: Class org.apache.hadoop.hive.ql.udf.generic.GenericUDFBridge can not access a member of class'. This was resolved by copying the JAR file into Hive LIB folder.
- Flume application was throwing Exception in thread "Twitter4J Async Dispatcher[0]" java.lang.OutOfMemoryError: Java heap space at java.util.Arrays.copyOf (Arrays.java:2367). This was resolved by uncommenting the statement from flume-conf.xml file.
- Field alias in pig script was throwing syntax error:'

```
>> Plan_ID != '' AND Segment_ID != ''AND CategoryCode=='1';

grunt>

grunt> PS_1 = FOREACH PREMAMNT_1 GENERATE Contract_ID, Plan_ID, Segment_ID,
CategoryCode,prem=premExtracter(Benefit);

2015-12-18 19:35:33,111 [main] ERROR org.apache.pig.tools.grunt.Grunt - ERROR 1200: <line 17, column 82>  Syntax error,
unexpected symbol at or near 'prem'

Details at logfile: /home/hduser/pig_1450488916101.log

grunt> PS = FILTER PS_1 BY prem !='none';

2015-12-18 19:35:44,290 [main] ERROR org.apache.pig.tools.grunt.Grunt - ERROR 1200: Pig script failed to parse:

<line 17, column 12> Undefined alias: PS_1

Details at logfile: /home/hduser/pig_1450488916101.log
```

This was resolved by using positional values instead of alias.

- Flume issue: Exception in thread "Twitter4J Async Dispatcher[0]" java.lang.OutOfMemoryError: Java heap space at java.util.Arrays.copyOf(Arrays.java:2367)
  I was getting this exception while I was trying to setup twitter messages into hdfs.

  It got resolved after I commented this line in flume-env.sh:
  export JAVA_OPTS="-Xms100m -Xmx1000m -Dcom.sun.management.jmxremote"

  The memory settings are there on flume conf file:
  # describe the channel TwitterAgent.channels.MemChannel.type=memory
  TwitterAgent.channels.MemChannel.capacity=10000
  TwitterAgent.channels.MemChannel.transactionCapacity=100

Appendix –A: Field Definitions for csv Files

Plan Info County Flat Files

These files are used to search plan information by county. The data is divided between two (2) zip files for different county fips code:

These files contain seventy five (75) fields and are designed to be searched individually:

1. Contract_ID: memo (-) - Lists the five (5) digit alpha-numeric code used to identify a contract.
2. Plan_ID: memo (-) - Lists the three (3) digit numeric code used to identify a particular plan.
3. Segment_ID: memo (-) - Lists the (1) digit numeric code used to identify a particular segment for a given plan.
4. Contract_Year: memo (-) - Lists the four (4) digit year in which the contract was created.
5. Org_Name: memo (-) - Lists the Organization granting the contract.
6. Plan_Name: memo (-) - Lists the name used for a particular plan.
7. Sp_Plan_Name: memo (-) - Lists the Spanish name used for a particular plan.
8. Geo_Name: memo (-) - Lists the name of the geographical area covered by a particular plan.
9. Tax_Status_Code: text (1) - Lists the numeric code used to identify the tax status of a particular plan.
10. Tax_status_desc: memo (-) - Lists the description of tax status (like, for-profit, nonprofit, etc.)
11. SP_tax_status_desc: memo (-) - Lists the Spanish description of tax status.
12. Plan_Type: long integer (4) - Lists the numeric code used to identify the category type of a particular plan.
13. Plan_type_desc: memo (-) - Lists the description of plan type.
14. Web_Address: memo (-) - Lists the internet address that corresponds to a particular plan.
15. PartD_Wb_Address: memo (-) - Lists the Part D web address.
16. Frmlry_Wbst_Adr: memo (-) - Lists the formulary website address of the contract.
17. Phrmcy_Wbst_Adr: memo (-) - Lists the pharmacy website address of the contract.
18. Fed_Approval_Status: memo (-) - Lists whether or not the plan is approved by Medicare.
19. Sp_Fed_Approval_Status: memo (-) - Lists in Spanish whether or not the plan is approved by Medicare.
20. Pos_Available_Flag: memo (-) - Lists whether or not a Point of Service plan is available.
21. Mail_Ordr_Avlblty: memo (-) - Indicates the mail order availability.
22. Cvrg_Gap_Ofrd: memo (-) - Indicates the coverage gap is available or not.
23. Cvrg_Gap_Ind: long integer (4) - Identifier of the coverage gap (0 – No coverage gap, 1 – Generic only, 2 – Generic and Preferred)
24. Cvrg_Gap_Desc: memo (-) - Lists the description of coverage gap.
25. Contract_Important_Note: memo (-) - Lists the important notes for the contract.
26. SP_Contract_Important_Note: memo (-) - Lists the Spanish important notes for the contract.
27. Plan_Important_Note: memo (-) - Lists any additional notes that exist regarding a particular plan.
28. SP_Plan_Important_Note: memo (-) - Lists in Spanish any additional notes that existregarding a particular plan.
29. Segment_Important_Note: memo (-) - Lists the important notes for the Segment.
30. SP_Segment_Important_Note: memo (-) - Lists the Spanish important notes for theSegment.
31. Legal_Entity_Name: memo (-) Lists the legal name of the contract.
32. Trade_Name: memo (-) - Lists the trademark name of the contract.
33. Network_English: memo (-) - Lists the physician network name in English.
34. Network_Spanish: memo (-) - Lists the physician network name in Spanish.
35. Contact_Person: memo (-)- Lists the name of the person or organization to be usedas a contact for a particular plan.
36. Street_Address: memo (-) - Lists the street address that corresponds to a particular plan.
37. City: memo (-) - Lists the city name that corresponds to a particular plan.
38. State_Code: memo (-) - Lists the alphabetic postal code used to identify each individual state. All fifty (50) states are listed, as well as:
    - DC for Washington D.C.
    - GU for Guam
    - PR for Puerto Rico
    - VI for Virgin Islands
39. Zip_Code: memo (-) - Lists the five (5) digit postal code used to identify each individual section of a county.
40. Email_prospective: memo (-) - Lists the contact email address for prospective recipients.
41. Local_Phone_prospective: memo (-) - Lists the local phone number for prospective members.

42. Tollfree_Phone_prospective: memo (-) - Lists the toll free phone number for prospective members.
43. Local_tty_prospective: memo (-) - Lists the local TTY phone number for prospective members.
44. Tollfree_tty_prospective: memo (-) - Lists the toll free TTY phone number for prospective members/
45. Email_Current: memo (-) - Lists the email for current members.
46. Local_Phone_Current: memo (-) - Lists the local phone number for current members.
47. Tollfree_Phone_Current: memo (-) - Lists the toll free phone number for current members.
48. Local_tty_current: memo (-) - Lists the local TTY phone number for current members.
49. Tollfree_tty_current: memo (-) - Lists the toll free TTY phone number for current members.
50. Contact_Person_pd: memo (-) - Lists the Part D contact person.
51. Street_Address_pd: memo (-) - Lists the street address for Part D contact.
52. City_pd: memo-) - Lists the City for Part D contact.
53. StateCode_pd: memo (-) - Lists the alphabetic postal code for Part D used to identify each individual state. All fifty (50) states are listed, as well as:
    - DC for Washington D.C.
    - GU for Guam
    - PR for Puerto Rico
    - VI for Virgin Islands
54. Zip_Code_pd: memo (-) - Lists the five (5) digit postal code for Part D contract used to identify each individual section of a county.
55. Email_prospective pd: memo (-) - Lists the email for prospective Part D members.
56. Local_Phone_prospective_pd: memo (-) - Lists the local phone number for prospective Part D members.
57. Tollfree_Phone_prospective_pd: memo (-) - Lists the toll free phone number for prospective Part D members.
58. Local_tty_prospective_pd: memo (-) - Lists the local TTY phone number for prospective Part D members.
59. Tollfree_tty_prospective_pd: memo (-) - Lists the toll free TTY phone number for prospective Part D members.
60. Email_Current_pd: memo (-) - Lists the email for current Part D members.
61. Local_Phone_Current_pd: memo (-) - Lists the local phone number for current Part D members.
62. Tollfree_Phone_Current_pd: memo (-) - Lists the toll free phone number for current Part D members.
63. Local_tty_current_pd: memo (-) - Lists the local TTY phone number for current Part D members.
64. Tollfree_tty_current_pd: memo (-) - Lists the toll free TTY phone number for current Part D members.
65. MA_PD_Indicator: memo (-) - Indicates the Medicare Advantage Part D plan.
66. PPO_PD_Indicator: memo (-) - Indicates the PPO Part D plan
67. Snp_Id: Text (1) - Indicates the identifier of Special needs plan.
68. Snp_Desc: memo (-) - Lists the description of Special needs plan.
69. Sp_Snp_Desc: memo (-) - Lists the Spanish description of Special needs plan.
70. Lis_100: memo (-) - Lists the low income subsidy premium for 100% subsidy level.
71. Lis_75: memo (-) - Lists the low income subsidy premium for 75% subsidy level.
72. Lis_50: memo (-) - Lists the low income subsidy premium for 50% subsidy level.
73. Lis_25: memo (-) - List the low income subsidy premium for 25% subsidy level.
74. Regional_Indicator: memo (-) - Indicates the regional contract.
75. CountyFIPSCode: long integer (4) - Lists the county FIPS code.

vwPlanServices.csv

1. Language
2. Contract_Year
3. Contract_ID
4. Plan_ID
5. Segment_ID
6. CategoryDescription
7. CategoryCode
8. Benefit
9. Package_Name
10. Package_ID
11. Sentences_Sort_Order

vwGeography.csv

1. StateCode
2. StateName
3. County_Name
4. CountyFIPSCode
5. Zip_Code

vwStarRating_SummaryScores

1. Contract_ID
2. Summary_Score
3. Star_Rating_Current
4. Star_Rating_Previous
5. Lang_dscrptn

Appendix – B: PIG LATIN SCRIPTS

---register_piggybank jar and define a function name for CSVExcelStorage()

```
REGISTER /home/hadoop/piggybank.jar;
DEFINE CSVLoader org.apache.pig.piggybank.storage.CSVExcelStorage();
```

----Load countyplan for PlanInfoCounty_FipsCodeMoreThan30000

```
PLAN1 = LOAD '/medicareplan/Plan1.csv' USING CSVLoader AS (contract_id:chararray, plan_id:chararray,
segment_id:chararray, contract_year:chararray, org_name:chararray, plan_name:chararray, sp_plan_name:chararray,
geo_name:chararray, tax_stus_cd:chararray, tax_status_desc:chararray, sp_tax_status_desc:chararray,
plan_type:chararray, plan_type_desc:chararray, web_address:chararray, partd_wb_adr:chararray,
frmlry_wbst_adr:chararray, phrmcy_wbst_adr:chararray, fed_approval_status:chararray,
sp_fed_approval_status:chararray, pos_available_flag:chararray, mail_ordr_avlblty:chararray, cvrg_gap_ofrd:chararray,
cvrg_gap_ind:chararray, cvrg_gap_desc:chararray, contract_important_note:chararray,
sp_contract_important_note:chararray, plan_important_note:chararray, sp_plan_important_note:chararray,
segment_important_note:chararray, sp_segment_important_note:chararray, legal_entity_name:chararray,
trade_name:chararray, network_english:chararray, network_spanish:chararray, contact_person:chararray,
street_address:chararray, city:chararray, state_code:chararray, zip_code:chararray, email_prospective:chararray,
local_phone_prospective:chararray, tollfree_phone_prospective:chararray, local_tty_prospective:chararray,
tollfree_tty_prospective:chararray, email_current:chararray, local_phone_current:chararray,
tollfree_phone_current:chararray, local_tty_current:chararray, tollfree_tty_current:chararray,
contact_person_pd:chararray, street_address_pd:chararray, city_pd:chararray,
state_code_pd:chararray, zip_code_pd:chararray, email_prospective_pd:chararray,
local_phone_prospective_pd:chararray, tollfree_phone_prospective_pd:chararray,
local_tty_prospective_pd:chararray, tollfree_tty_prospective_pd:chararray, email_current_pd:chararray,
local_phone_current_pd:chararray, tollfree_phone_current_pd:chararray, local_tty_current_pd:chararray,
tollfree_tty_current_pd:chararray, ma_pd_indicator:chararray, ppo_pd_indicator:chararray, snp_id:chararray,
snp_desc:chararray, sp_snp_desc:chararray, lis_100:chararray, lis_75:chararray, lis_50:chararray, lis_25:chararray,
regional_indicator:chararray, CountyFIPSCode:chararray);
```

----Clean up the data using filter

```
PC1 = FILTER PLAN1 BY contract_id != 'contract_id' AND contract_id !='' AND plan_id != '' AND segment_id != '' AND
plan_name != '' AND CountyFIPSCode != '';
```

----Select required Fields

```
PL1 = FOREACH PC1 GENERATE contract_id, plan_id, segment_id, org_name, plan_name, plan_type,CountyFIPSCode;
```

----Perform the same functions for ***PlanInfoCounty_FipsCodeLessThan30000*** data

```
PLAN2 = LOAD '/medicareplan/Plan2.csv' USING CSVLoader AS (contract_id:chararray, plan_id:chararray,
segment_id:chararray, contract_year:chararray, org_name:chararray, plan_name:chararray, sp_plan_name:chararray,
geo_name:chararray, tax_stus_cd:chararray, tax_status_desc:chararray, sp_tax_status_desc:chararray,
plan_type:chararray, plan_type_desc:chararray, web_address:chararray, partd_wb_adr:chararray,
frmlry_wbst_adr:chararray, phrmcy_wbst_adr:chararray, fed_approval_status:chararray,
sp_fed_approval_status:chararray, pos_available_flag:chararray, mail_ordr_avlblty:chararray, cvrg_gap_ofrd:chararray,
cvrg_gap_ind:chararray, cvrg_gap_desc:chararray, contract_important_note:chararray,
sp_contract_important_note:chararray, plan_important_note:chararray, sp_plan_important_note:chararray,
segment_important_note:chararray, sp_segment_important_note:chararray, legal_entity_name:chararray,
trade_name:chararray, network_english:chararray, network_spanish:chararray, contact_person:chararray,
street_address:chararray, city:chararray, state_code:chararray, zip_code:chararray, email_prospective:chararray,
local_phone_prospective:chararray, tollfree_phone_prospective:chararray, local_tty_prospective:chararray,
tollfree_tty_prospective:chararray, email_current:chararray, local_phone_current:chararray,
tollfree_phone_current:chararray, local_tty_current:chararray, tollfree_tty_current:chararray,
contact_person_pd:chararray, street_address_pd:chararray, city_pd:chararray,
state_code_pd:chararray, zip_code_pd:chararray, email_prospective_pd:chararray,
local_phone_prospective_pd:chararray, tollfree_phone_prospective_pd:chararray,
local_tty_prospective_pd:chararray, tollfree_tty_prospective_pd:chararray, email_current_pd:chararray,
local_phone_current_pd:chararray, tollfree_phone_current_pd:chararray, local_tty_current_pd:chararray,
tollfree_tty_current_pd:chararray, ma_pd_indicator:chararray, ppo_pd_indicator:chararray, snp_id:chararray,
snp_desc:chararray, sp_snp_desc:chararray, lis_100:chararray, lis_75:chararray, lis_50:chararray, lis_25:chararray,
regional_indicator:chararray, CountyFIPSCode:chararray);


PC2 = FILTER PLAN2 BY contract_id != 'contract_id' AND contract_id !='' AND  plan_id != '' AND segment_id != '' AND
plan_name != '' AND CountyFIPSCode != '';

PL2 = FOREACH PC2 GENERATE contract_id, plan_id, segment_id, org_name, plan_name, plan_type,CountyFIPSCode;
```

----Combine the two countyplan data using Union and then store the data on HDFS

```
PLAN  = UNION PL1, PL2;

STORE PLAN INTO '/medicareplan/countybucket' using PigStorage(',','-schema');
```

----Load Plan service data from vwPlanServices.csv

```
PLANSERVICE = LOAD '/medicareplan/vwplanservices.csv' USING CSVLoader AS
(Language:chararray, Contract_Year:chararray, Contract_ID:chararray,
Plan_ID:chararray,
Segment_ID:chararray,
CategoryDescription:chararray,
CategoryCode:chararray, Benefit:chararray );
PS_F = FILTER PLANSERVICE BY Language == 'English' AND Contract_ID != 'Contract_ID' AND Contract_ID !=" AND
Plan_ID != " AND Segment_ID != '';
PS= FOREACH PS_F GENERATE Contract_ID, Plan_ID, Segment_ID,
CategoryDescription, CategoryCode,Benefit;
STORE PS INTO '/medicareplan/planservice' using PigStorage(',','-schema');
```

----Load county name data from vwGeography.csv

```
COUNTY_NAME = LOAD '/medicareplan/vwgeography.csv' USING CSVLoader AS
(StateCode:chararray,  StateName:chararray,  CountyName:chararray,  FIPSCode:chararray,
zip_code:chararray);
CNF = FILTER COUNTY_NAME BY StateCode!= 'StateCode' AND CountyName != " AND FIPSCode != '';
CN = FOREACH CNF GENERATE FIPSCode, CountyName;
STORE CN INTO '/medicareplan/countynames' using PigStorage(',','-schema');
```

----Load Star Rating Summary data from vwStarRating_SummaryScores

```
STARRATE = LOAD '/medicareplan/vwstarsummary.csv' USING CSVLoader AS
(Contract_ID:chararray,  Summary_Score:chararray, Star_Rating_Current:chararray,
Star_Rating_Previous: chararray, lang_dscrptn: chararray);
SR_F = FILTER STARRATE BY lang_dscrptn == 'English' AND Contract_ID != 'Contract_ID'
AND Contract_ID !='';
SR_B = FOREACH SR_F GENERATE Contract_ID, Summary_Score, Star_Rating_Current,Star_Rating_Previous;STORE SR_B
INTO '/medicareplan/starsummary' using PigStorage(',','-schema');
```

Appendix-C: Hive Queries

-/* Hive DDL Statements */

-/*Define Medicare Database*/

```
use medicare;
```

-/*Add my JAR file to Hive Library and define UDFs*/

```
add jar myhiveudfs.jar;
CREATE TEMPORARY FUNCTION premcalc as 'com.rama.myhiveudfs.PremCalcUDF';
CREATE TEMPORARY FUNCTION copaycalc as 'com.rama.myhiveudfs.CopayCalcUDF';
CREATE TEMPORARY FUNCTION coinscalc as 'com.rama.myhiveudfs.CoinsCalcUDF';
```

-/*Set Hive Optimization criterias*/

```
set hive.enforce.bucketing=true;
set hive.optimize.bucketmapjoin=true;
set hive.exec.parallel=true;
```

- /* Load Star Rating summary table from Pig into Hive */

```
CREATE TABLE starratetable
(ContractId  STRING, summary_score STRING,  star_rating_Current STRING, star_rating_previous STRING)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
LOCATION '/medicareplan/starsummary';
```

- /* Load County table from Pig into Hive */

```
CREATE TABLE countytable
(contract_id  STRING, plan_id  STRING,  segment_id  STRING,  org_name  STRING,  plan_name STRING,
 plan_type STRING, countyFIPcode STRING)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
LOCATION '/medicareplan/county';
```

- /* Load County table with bucket clustering using the field CountyFIPcode: from Pig into Hive */

```
CREATE TABLE countybuckettable
(contract_id STRING, plan_id STRING, segment_id STRING, org_name STRING, plan_name STRING,
 plan_type STRING, countyFIPcode STRING)
CLUSTERED BY (CountyFIPcode) INTO 64 BUCKETS
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
LOCATION '/medicareplan/countybucket';
```

- /* Load planservice table from Pig into Hive */

```
CREATE TABLE planservicetable
(contractid STRING, planid STRING, segmentid STRING, category STRING, categoryCode STRING,
 benefit STRING)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
LOCATION '/medicareplan/planservice';
```

- /* Load countynames table from Pig into Hive */

```
CREATE TABLE countynamestable
(FIPScode STRING, countyname STRING)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
LOCATION '/medicareplan/countynames';
```

==========

-/*Hive DML Statements*/

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

```
SELECT CountyFIPcode,org_name, plan_name,max(premcalc(benefit)) as prem
FROM countytable
JOIN planservicetable
ON countytable.contract_id = planservicetable.contractid AND countytable.plan_id = planservicetable.planid
WHERE CategoryCode="1" AND (instr(benefit,"monthly premium")>0 OR instr(benefit,"per month")>0 OR instr(benefit,"per year")>0)
AND countyFIPcode = "97701"
GROUP BY countyFIPcode,org_name, plan_name
SORT BYprem
LIMIT 5;
```

-/*2. To find plans that have highest co-pays for doctors in a given county*/

```
SELECT countyFIPcode,org_name, plan_name, max(copaycalc(benefit)) as copay
FROM countytable
JOIN planservicetable
ON countytable.contract_id = planservicetable.contractid AND countytable.plan_id = planservicetable.planid
WHERE CategoryCode="10" AND countyFIPcode = "97701"
GROUP BY countyFIPcode,org_name, plan_name
ORDER BYcopay
LIMIT 5;
```

-/*3. To compare plans based on features like plans that offer free ambulance services*/

```
SELECT countyFIPcode,org_name, plan_name
from countytable
Join planservicetable
ON countytable.contract_id = planservicetable.contractid AND countytable.plan_id = planservicetable.planid
where CategoryCode="5" AND copaycalc(benefit)="0"
GROUP BY countyFIPcode,org_name, plan_name
ORDER BY countyFIPcode
LIMIT 5;
```

-/*4. To compare plans based on features like the benefits available for diabetes under specific plan*/

```
SELECT  countyFIPcode,org_name, plan_name
from countytable
Join planservicetable
ON countytable.contract_id  = planservicetable.contractid  AND countytable.plan_id = planservicetable.planid
where countyFIPcode = "25003" and (CategoryCode="8"  OR instr(benefit,"diabetes")>0)
GROUP BY countyFIPcode,org_name, plan_name
ORDER BY countyFIPcode
LIMIT 5;
```

-/*5. To compare plan benefits on diabetes and mental healthcare offered by all companies in a particular*/ ----------county

```
SELECT  countyFIPcode,org_name, plan_name
from countytable
Join planservicetable
ON countytable.contract_id  = planservicetable.contractid  AND countytable.plan_id = planservicetable.planid
where countyFIPcode = "25003" and (CategoryCode="8" OR CategoryCode="16" OR instr(benefit,"diabetes")>0
OR instr(benefit,"mental")>0)
GROUP BY countyFIPcode,org_name, plan_name
ORDER BY countyFIPcode
LIMIT 5;
```

==========

Hive queries for processing twitter messages:

-/*6. To find out plans provided by organizations having maximum twitter presence*/;

```
-/* Add serde jar to hive*/;
add jar hive-serdes-1.0-SNAPSHOT.jar;
set hive.support.sql11.reserved.keywords=false;
```

-/*Create Hive Table Structure for twitter messages*/;

```
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 '/twitter/twitter';
```

-/* Create a Hive Table for Plan Provider Names*/

```
CREATE table words (word STRING);

INSERT INTO words

VALUES

('Westmoreland'),('WellCare'),('VillageCareMAX'),('SummaCare'),('StayWell'),('SelectCare'),('PrimeWest'),('PacificSource'),('OneCare'),('MedStar'),('KelseyCare'),('InovaCares'),('IliniCare'),('IEHP'),('HUMANA'),('HealthSun'),('HealthSpring'),('HealthPlus'),('Healthfirst'),('GLOBALHEALTH'),('CDPHP'),('CareSource'),('CarePlus'),('CarePartners'),('CareOregon'),('CareMore'),('Care1st'),('BlueCross'),('BlueChoice'),('BlueCare'),('ArchCare'),('AmeriHealth'),('Amerigroup Healthcare'),('AltaMed'),('AlphaCare'),('AllCare'),('Aetna'),('Advicare');
```

-/*Create a Hive Table and insert the twitter message text field from table tweets*/

```
CREATE table tweettexts (texts STRING);

INSERT INTO tweettexts

SELECT  text FROM  tweets;
```

-/*Create a Hive Table and insert the Plan provider names and the number of text having this name in the */ --

-/*message text field from table tweets*/

```
CREATE table tweetpresence(Planprovider STRING, presence DOUBLE);

INSERT INTO  tweetpresence

SELECT  word, COUNT(texts) FROM  words, tweettexts WHERE instr(texts,word)>0

GROUP BY word;
```

-/* Select 5 plan providers having highest reference in the twitter messages*/

```
SELECT  Planprovider, presence FROM  tweetpresence ORDER BYpresence desc limit 5;
```

Appendix-D: Flume Configuration

```
# name of the components
TwitterAgent.sources=Twitter
TwitterAgent.channels=MemChannel
TwitterAgent.sinks=HDFS
# describe and configure the source
TwitterAgent.sources.Twitter.type = com.cloudera.flume.source.TwitterSource
TwitterAgent.sources.Twitter.channels=MemChannel

TwitterAgent.sources.Twitter.consumerKey = …
TwitterAgent.sources.Twitter.consumerSecret = …
TwitterAgent.sources.Twitter.accessToken =  …
TwitterAgent.sources.Twitter.accessTokenSecret =  …

# Twitter handles to search
TwitterAgent.sources.Twitter.keywords=
medicare,Westmoreland,WellCare,VillageCareMAX,UCare's,SummaCare,StayWell,SelectCare,PrimeWest,P
acificSource,OneCare,MedStar,Medicare,KelseyCare,InovaCares,lliniCare,IEHP,HUMANA,Humana,HealthS
un,HealthSpring,HealthPlus,Healthfirst,GLOBALHEALTH,CDPHP,CareSource,CarePlus,CarePartners,Care
Oregon,CareMore,Care1st,BlueCross,BlueChoice,BlueCare,ArchCare,AmeriHealth,Amerigroup
Healthcare,AltaMed,AlphaCare,AllCare,Aetna,Advicare
# describe the sink
TwitterAgent.sinks.HDFS.channel=MemChannel
TwitterAgent.sinks.HDFS.type=hdfs
TwitterAgent.sinks.HDFS.hdfs.path=hdfs://localhost:9000/flumegrab/twitter
TwitterAgent.sinks.HDFS.hdfs.fileType=DataStream
TwitterAgent.sinks.HDFS.hdfs.writeformat=Text
TwitterAgent.sinks.HDFS.hdfs.batchSize=100
TwitterAgent.sinks.HDFS.hdfs.rollSize=0
TwitterAgent.sinks.HDFS.hdfs.rollCount=10000
TwitterAgent.sinks.HDFS.hdfs.rollInterval=600
# describe the channel
TwitterAgent.channels.MemChannel.type=memory
TwitterAgent.channels.MemChannel.capacity=10000
TwitterAgent.channels.MemChannel.transactionCapacity=100
```

Appendix-E

Screen Shot -1: With subqueries/temporary tables and without Bucket Clustering

```
hive> SELECT contract_id,countyname,org_name, plan_name, max(premcalc(benefit)) prem,star_rating_Current, star_rating_previous
    > FROM countytable
    > join (Select FIPScode,countyname from countynamestable where FIPScode = "25003" Limit 1) as cn
    > ON (countytable.countyFIPcode= cn.FIPScode)
    > JOIN (select planid,contractid,benefit from planservicetable
    > where CategoryCode="1" AND (instr(benefit,"monthly premium")>0 OR instr(benefit,"per month")>0  OR instr(benefit,"per year")>0))
as ps
    > on (ps.planid = countytable.plan_id and ps.contractid =countytable.contract_id)
    > left outer join (select ContractId, star_rating_Current, star_rating_previous
    > from starratetable where summary_score = 'Overall Star Rating') as sr
    > ON (sr.ContractId = countytable.contract_id)
    > Group by contract_id,countyname,org_name, plan_name,star_rating_Current, star_rating_previous
    > order by prem
    > LIMIT 5;
Query ID = hadoop_20160129194526_ee4a707f-916e-477f-83f4-afc4d86c9bee
Total jobs = 9
Launching Job 1 out of 9
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1454117877134_0001, Tracking URL = http://hadoop:8088/proxy/application_1454117877134_0001/
Kill Command = /usr/local/hadoop/bin/hadoop job  -kill job_1454117877134_0001
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2016-01-29 19:45:59,961 Stage-1 map = 0%,  reduce = 0%
2016-01-29 19:46:17,003 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 3.64 sec
2016-01-29 19:46:31,848 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 5.98 sec
MapReduce Total cumulative CPU time: 5 seconds 980 msec
Ended Job = job_1454117877134_0001
Stage-20 is selected by condition resolver.
Stage-21 is filtered out by condition resolver.
Stage-2 is filtered out by condition resolver.
Execution log at: /tmp/hadoop/hadoop_20160129194526_ee4a707f-916e-477f-83f4-afc4d86c9bee.log
2016-01-29 19:46:46     Starting to launch local task to process map join;      maximum memory = 518979584
```

2016-01-29 19:46:48 Dump the side-table for tag: 1 with group count: 1 into file: file:/tmp/hadoop/0e60428c-a2f1-4c08-b9c2-c80f083a0d56/hive_2016-01-29_19-45-26_591_7784149470723285913-1/-local-10015/HashTable-Stage-14/MapJoin-mapfile31--.hashtable

2016-01-29 19:46:48 Uploaded 1 File to: file:/tmp/hadoop/0e60428c-a2f1-4c08-b9c2-c80f083a0d56/hive_2016-01-29_19-45-26_591_7784149470723285913-1/-local-10015/HashTable-Stage-14/MapJoin-mapfile31--.hashtable (294 bytes)

2016-01-29 19:46:48 End of local task; Time Taken: 1.95 sec.

Execution completed successfully

MapredLocal task succeeded

Launching Job 3 out of 9

Number of reduce tasks is set to 0 since there's no reduce operator

Starting Job = job_1454117877134_0002, Tracking URL = http://hadoop:8088/proxy/application_1454117877134_0002/

Kill Command = /usr/local/hadoop/bin/hadoop job -kill job_1454117877134_0002

Hadoop job information for Stage-14: number of mappers: 1; number of reducers: 0

2016-01-29 19:47:05,050 Stage-14 map = 0%, reduce = 0%

2016-01-29 19:47:20,589 Stage-14 map = 35%, reduce = 0%, Cumulative CPU 4.67 sec

2016-01-29 19:47:23,922 Stage-14 map = 83%, reduce = 0%, Cumulative CPU 6.39 sec

2016-01-29 19:47:24,996 Stage-14 map = 100%, reduce = 0%, Cumulative CPU 7.01 sec

MapReduce Total cumulative CPU time: 7 seconds 10 msec

Ended Job = job_1454117877134_0002

Stage-18 is filtered out by condition resolver.

Stage-19 is selected by condition resolver.

Stage-3 is filtered out by condition resolver.

Execution log at: /tmp/hadoop/hadoop_20160129194526_ee4a707f-916e-477f-83f4-afc4d86c9bee.log

2016-01-29 19:47:38 Starting to launch local task to process map join; maximum memory = 518979584

2016-01-29 19:47:40 Dump the side-table for tag: 0 with group count: 352 into file: file:/tmp/hadoop/0e60428c-a2f1-4c08-b9c2-c80f083a0d56/hive_2016-01-29_19-45-26_591_7784149470723285913-1/-local-10013/HashTable-Stage-12/MapJoin-mapfile20--.hashtable

2016-01-29 19:47:40 Uploaded 1 File to: file:/tmp/hadoop/0e60428c-a2f1-4c08-b9c2-c80f083a0d56/hive_2016-01-29_19-45-26_591_7784149470723285913-1/-local-10013/HashTable-Stage-12/MapJoin-mapfile20--.hashtable (35206 bytes)

2016-01-29 19:47:40 End of local task; Time Taken: 2.066 sec.

Execution completed successfully

MapredLocal task succeeded

Launching Job 5 out of 9

Number of reduce tasks is set to 0 since there's no reduce operator

Starting Job = job_1454117877134_0003, Tracking URL = http://hadoop:8088/proxy/application_1454117877134_0003/

Kill Command = /usr/local/hadoop/bin/hadoop job -kill job_1454117877134_0003

Hadoop job information for Stage-12: number of mappers: 1; number of reducers: 0

2016-01-29 19:47:55,039 Stage-12 map = 0%, reduce = 0%

2016-01-29 19:48:13,067 Stage-12 map = 100%, reduce = 0%, Cumulative CPU 5.85 sec

MapReduce Total cumulative CPU time: 5 seconds 850 msec

Ended Job = job_1454117877134_0003

Execution log at: /tmp/hadoop/hadoop_20160129194526_ee4a707f-916e-477f-83f4-afc4d86c9bee.log

2016-01-29 19:48:24 Starting to launch local task to process map join; maximum memory = 518979584

2016-01-29 19:48:27 Dump the side-table for tag: 1 with group count: 558 into file: file:/tmp/hadoop/0e60428c-a2f1-4c08-b9c2-c80f083a0d56/hive_2016-01-29_19-45-26_591_7784149470723285913-1/-local-10009/HashTable-Stage-5/MapJoin-mapfile01--.hashtable

2016-01-29 19:48:27 Uploaded 1 File to: file:/tmp/hadoop/0e60428c-a2f1-4c08-b9c2-c80f083a0d56/hive_2016-01-29_19-45-26_591_7784149470723285913-1/-local-10009/HashTable-Stage-5/MapJoin-mapfile01--.hashtable (36975 bytes)

2016-01-29 19:48:27 End of local task; Time Taken: 2.963 sec.

Execution completed successfully

MapredLocal task succeeded

Launching Job 6 out of 9

Number of reduce tasks not specified. Estimated from input data size: 1

In order to change the average load for a reducer (in bytes):

set hive.exec.reducers.bytes.per.reducer=<number>

In order to limit the maximum number of reducers:

set hive.exec.reducers.max=<number>

In order to set a constant number of reducers:

set mapreduce.job.reduces=<number>

Starting Job = job_1454117877134_0004, Tracking URL = http://hadoop:8088/proxy/application_1454117877134_0004/

Kill Command = /usr/local/hadoop/bin/hadoop job -kill job_1454117877134_0004

Hadoop job information for Stage-5: number of mappers: 1; number of reducers: 1

2016-01-29 19:48:43,313 Stage-5 map = 0%, reduce = 0%

2016-01-29 19:48:56,762 Stage-5 map = 100%, reduce = 0%, Cumulative CPU 3.2 sec

2016-01-29 19:49:10,038 Stage-5 map = 100%, reduce = 100%, Cumulative CPU 5.31 sec

MapReduce Total cumulative CPU time: 5 seconds 310 msec

Ended Job = job_1454117877134_0004

Launching Job 7 out of 9

Number of reduce tasks determined at compile time: 1

In order to change the average load for a reducer (in bytes):

set hive.exec.reducers.bytes.per.reducer=<number>

In order to limit the maximum number of reducers:

set hive.exec.reducers.max=<number>

In order to set a constant number of reducers:

set mapreduce.job.reduces=<number>

Starting Job = job_1454117877134_0005, Tracking URL = http://hadoop:8088/proxy/application_1454117877134_0005/

Kill Command = /usr/local/hadoop/bin/hadoop job -kill job_1454117877134_0005

Hadoop job information for Stage-6: number of mappers: 1; number of reducers: 1

2016-01-29 19:49:28,740 Stage-6 map = 0%, reduce = 0%

2016-01-29 19:49:39,721 Stage-6 map = 100%, reduce = 0%, Cumulative CPU 1.52 sec

2016-01-29 19:49:51,813 Stage-6 map = 100%, reduce = 100%, Cumulative CPU 3.7 sec

MapReduce Total cumulative CPU time: 3 seconds 700 msec

Ended Job = job_1454117877134_0005

MapReduce Jobs Launched:

Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 5.98 sec HDFS Read: 14295 HDFS Write: 123 SUCCESS

```
Stage-Stage-14: Map: 1   Cumulative CPU: 7.01 sec   HDFS Read: 95612673 HDFS Write: 34842 SUCCESS
Stage-Stage-12: Map: 1   Cumulative CPU: 5.85 sec   HDFS Read: 53734676 HDFS Write: 5012 SUCCESS
Stage-Stage-5: Map: 1  Reduce: 1   Cumulative CPU: 5.31 sec   HDFS Read: 20226 HDFS Write: 1172 SUCCESS
Stage-Stage-6: Map: 1  Reduce: 1   Cumulative CPU: 3.7 sec   HDFS Read: 7075 HDFS Write: 581 SUCCESS
Total MapReduce CPU Time Spent: 27 seconds 850 msec
OK
H8578   BERKSHIRE       HNE Medicare Advantage Plans    HNE Medicare Basic No Rx (HMO)  123.9   4 out of 5 stars        4.5 out of 5 stars
H8578   BERKSHIRE       HNE Medicare Advantage Plans    HNE Medicare Value (HMO)        124.9   4 out of 5 stars        4.5 out of 5 stars
R7444   BERKSHIRE       UnitedHealthcare        AARP MedicareComplete Choice (Regional PPO)     144.9   4 out of 5 stars        3.5 out of 5 stars
H8578   BERKSHIRE       HNE Medicare Advantage Plans    HNE Medicare Basic (HMO)        179.9   4 out of 5 stars        4.5 out of 5 stars
H8578   BERKSHIRE       HNE Medicare Advantage Plans    HNE Medicare Premium No Rx (HMO)        193.9   4 out of 5 stars        4.5 out of 5 stars
Time taken: 266.394 seconds, Fetched: 5 row(s)
```

Screen Shot -2: Without subqueries/temporary tables and without Bucket Clustering

```
hive> SELECT contract_id,countyname,org_name, plan_name, max(premcalc(benefit)) prem,star_rating_Current, star_rating_previous
    > FROM countytable
    > join countynamestable ON (countynamestable.FIPScode = countytable.countyFIPcode)
    > JOIN planservicetable on (planservicetable.planid = countytable.plan_id and countytable.contract_id  = planservicetable.contractid)
    > left outer join starratetable ON (countytable.contract_id  = starratetable.ContractId)
    > where countytable.countyFIPcode = "25003"
    > AND CategoryCode="1" AND (instr(benefit,"monthly premium")>0 OR instr(benefit,"per month")>0  OR instr(benefit,"per year")>0)
    > AND summary_score = 'Overall Star Rating'
    > Group by contract_id,countyname,org_name, plan_name,star_rating_Current, star_rating_previous
    > order by prem
    > LIMIT 5;
Query ID = hadoop_20160129202658_dc2617a5-4a39-4f28-8d0e-12477a550d74
Total jobs = 6
Execution log at: /tmp/hadoop/hadoop_20160129202658_dc2617a5-4a39-4f28-8d0e-12477a550d74.log
2016-01-29 20:27:09     Starting to launch local task to process map join;      maximum memory = 518979584
2016-01-29 20:27:11     Dump the side-table for tag: 1 with group count: 1 into file: file:/tmp/hadoop/0831be3d-884e-435f-81c9-98ee187a1beb/hive_2016-01-29_20-26-58_834_8418887664488278120-1/-local-10014/HashTable-Stage-13/MapJoin-mapfile31--.hashtable
2016-01-29 20:27:11     Uploaded 1 File to: file:/tmp/hadoop/0831be3d-884e-435f-81c9-98ee187a1beb/hive_2016-01-29_20-26-58_834_8418887664488278120-1/-local-10014/HashTable-Stage-13/MapJoin-mapfile31--.hashtable (867 bytes)
2016-01-29 20:27:11     End of local task; Time Taken: 2.0 sec.
Execution completed successfully
MapredLocal task succeeded
Launching Job 1 out of 6
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1454117877134_0006, Tracking URL = http://hadoop:8088/proxy/application_1454117877134_0006/
Kill Command = /usr/local/hadoop/bin/hadoop job  -kill job_1454117877134_0006
```

Hadoop job information for Stage-13: number of mappers: 1; number of reducers: 0

2016-01-29 20:27:24,180 Stage-13 map = 0%, reduce = 0%

2016-01-29 20:27:37,380 Stage-13 map = 100%, reduce = 0%, Cumulative CPU 5.02 sec

MapReduce Total cumulative CPU time: 5 seconds 20 msec

Ended Job = job_1454117877134_0006

Stage-15 is filtered out by condition resolver.

Stage-16 is selected by condition resolver.

Stage-2 is filtered out by condition resolver.

Execution log at: /tmp/hadoop/hadoop_20160129202658_dc2617a5-4a39-4f28-8d0e-12477a550d74.log

2016-01-29 20:27:46 Starting to launch local task to process map join; maximum memory = 518979584

2016-01-29 20:27:48 Dump the side-table for tag: 0 with group count: 352 into file: file:/tmp/hadoop/0831be3d-884e-435f-81c9-98ee187a1beb/hive_2016-01-29_20-26-58_834_8418887664488278120-1/-local-10012/HashTable-Stage-11/MapJoin-mapfile20--.hashtable

2016-01-29 20:27:49 Uploaded 1 File to: file:/tmp/hadoop/0831be3d-884e-435f-81c9-98ee187a1beb/hive_2016-01-29_20-26-58_834_8418887664488278120-1/-local-10012/HashTable-Stage-11/MapJoin-mapfile20--.hashtable (1054074 bytes)

2016-01-29 20:27:49 End of local task; Time Taken: 2.276 sec.

Execution completed successfully

MapredLocal task succeeded

Launching Job 3 out of 6

Number of reduce tasks is set to 0 since there's no reduce operator

Starting Job = job_1454117877134_0007, Tracking URL = http://hadoop:8088/proxy/application_1454117877134_0007/

Kill Command = /usr/local/hadoop/bin/hadoop job -kill job_1454117877134_0007

Hadoop job information for Stage-11: number of mappers: 1; number of reducers: 0

2016-01-29 20:27:59,296 Stage-11 map = 0%, reduce = 0%

2016-01-29 20:28:11,060 Stage-11 map = 100%, reduce = 0%, Cumulative CPU 3.98 sec

MapReduce Total cumulative CPU time: 3 seconds 980 msec

Ended Job = job_1454117877134_0007

Execution log at: /tmp/hadoop/hadoop_20160129202658_dc2617a5-4a39-4f28-8d0e-12477a550d74.log

2016-01-29 20:28:19 Starting to launch local task to process map join; maximum memory = 518979584

2016-01-29 20:28:21 Dump the side-table for tag: 1 with group count: 642 into file: file:/tmp/hadoop/0831be3d-884e-435f-81c9-98ee187a1beb/hive_2016-01-29_20-26-58_834_8418887664488278120-1/-local-10008/HashTable-Stage-4/MapJoin-mapfile01--.hashtable

2016-01-29 20:28:21 Uploaded 1 File to: file:/tmp/hadoop/0831be3d-884e-435f-81c9-98ee187a1beb/hive_2016-01-29_20-26-58_834_8418887664488278120-1/-local-10008/HashTable-Stage-4/MapJoin-mapfile01--.hashtable (157660 bytes)

2016-01-29 20:28:21 End of local task; Time Taken: 1.453 sec.

Execution completed successfully

MapredLocal task succeeded

Launching Job 4 out of 6

Number of reduce tasks not specified. Estimated from input data size: 1

In order to change the average load for a reducer (in bytes):

 set hive.exec.reducers.bytes.per.reducer=<number>

In order to limit the maximum number of reducers:

 set hive.exec.reducers.max=<number>

In order to set a constant number of reducers:

 set mapreduce.job.reduces=<number>

Starting Job = job_1454117877134_0008, Tracking URL = http://hadoop:8088/proxy/application_1454117877134_0008/

Kill Command = /usr/local/hadoop/bin/hadoop job -kill job_1454117877134_0008

Hadoop job information for Stage-4: number of mappers: 1; number of reducers: 1

```
2016-01-29 20:28:32,724 Stage-4 map = 0%,  reduce = 0%
2016-01-29 20:28:41,340 Stage-4 map = 100%,  reduce = 0%, Cumulative CPU 2.28 sec
2016-01-29 20:28:50,909 Stage-4 map = 100%,  reduce = 100%, Cumulative CPU 3.8 sec
MapReduce Total cumulative CPU time: 3 seconds 800 msec
Ended Job = job_1454117877134_0008
Launching Job 5 out of 6
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1454117877134_0009, Tracking URL = http://hadoop:8088/proxy/application_1454117877134_0009/
Kill Command = /usr/local/hadoop/bin/hadoop job  -kill job_1454117877134_0009
Hadoop job information for Stage-5: number of mappers: 1; number of reducers: 1
2016-01-29 20:29:05,660 Stage-5 map = 0%,  reduce = 0%
2016-01-29 20:29:14,272 Stage-5 map = 100%,  reduce = 0%, Cumulative CPU 1.06 sec
2016-01-29 20:29:22,939 Stage-5 map = 100%,  reduce = 100%, Cumulative CPU 2.52 sec
MapReduce Total cumulative CPU time: 2 seconds 520 msec
Ended Job = job_1454117877134_0009
MapReduce Jobs Launched:
Stage-Stage-13: Map: 1   Cumulative CPU: 5.02 sec   HDFS Read: 95613168 HDFS Write: 1355810 SUCCESS
Stage-Stage-11: Map: 1   Cumulative CPU: 3.98 sec   HDFS Read: 53734315 HDFS Write: 192000 SUCCESS
Stage-Stage-4: Map: 1  Reduce: 1   Cumulative CPU: 3.8 sec   HDFS Read: 208919 HDFS Write: 1172 SUCCESS
Stage-Stage-5: Map: 1  Reduce: 1   Cumulative CPU: 2.52 sec   HDFS Read: 7075 HDFS Write: 581 SUCCESS
Total MapReduce CPU Time Spent: 15 seconds 320 msec
OK
H8578   BERKSHIRE       HNE Medicare Advantage Plans    HNE Medicare Basic No Rx (HMO)  123.9   4 out of 5 stars        4.5 out of 5 stars
H8578   BERKSHIRE       HNE Medicare Advantage Plans    HNE Medicare Value (HMO)        124.9   4 out of 5 stars        4.5 out of 5 stars
R7444   BERKSHIRE       UnitedHealthcare        AARP MedicareComplete Choice (Regional PPO)     144.9   4 out of 5 stars        3.5 out of 5 stars
H8578   BERKSHIRE       HNE Medicare Advantage Plans    HNE Medicare Basic (HMO)        179.9   4 out of 5 stars        4.5 out of 5 stars
H8578   BERKSHIRE       HNE Medicare Advantage Plans    HNE Medicare Premium No Rx (HMO)        193.9   4 out of 5 stars        4.5 out of 5 stars
Time taken: 147.379 seconds, Fetched: 5 row(s)
```

Screen Shot -3: With subqueries/temporary tables and with Bucket Clustering

```
hive> SELECT contract_id,countyname,org_name, plan_name, max(premcalc(benefit)) prem,star_rating_Current, star_rating_previous
    > FROM countytable
    > join (Select FIPScode,countyname from countynamestable where FIPScode = "25003" Limit 1) as cn
    > ON (countytable.countyFIPcode= cn.FIPScode)
    > JOIN (select planid,contractid,benefit from planservicetable
    > where CategoryCode="1" AND (instr(benefit,"monthly premium")>0 OR instr(benefit,"per month")>0  OR instr(benefit,"per year")>0)) as ps
    > on (ps.planid = countytable.plan_id and ps.contractid =countytable.contract_id)
```

```
   > left outer join (select ContractId, star_rating_Current, star_rating_previous
   > from starratetable where summary_score = 'Overall Star Rating') as sr
   > ON (sr.ContractId = countytable.contract_id)
   > Group by contract_id,countyname,org_name, plan_name,star_rating_Current, star_rating_previous
   > order by prem
   > LIMIT 5;
Query ID = hadoop_20160129203507_879eb87a-16c8-4c13-9457-6f354efffc58
Total jobs = 9
Launching Job 1 out of 9
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1454117877134_0010, Tracking URL = http://hadoop:8088/proxy/application_1454117877134_0010/
Kill Command = /usr/local/hadoop/bin/hadoop job  -kill job_1454117877134_0010
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2016-01-29 20:35:23,555 Stage-1 map = 0%,  reduce = 0%
2016-01-29 20:35:32,341 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.8 sec
2016-01-29 20:35:42,196 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 3.18 sec
MapReduce Total cumulative CPU time: 3 seconds 180 msec
Ended Job = job_1454117877134_0010
Stage-20 is selected by condition resolver.
Stage-21 is filtered out by condition resolver.
Stage-2 is filtered out by condition resolver.
Execution log at: /tmp/hadoop/hadoop_20160129203507_879eb87a-16c8-4c13-9457-6f354efffc58.log
2016-01-29 20:35:52     Starting to launch local task to process map join;      maximum memory = 518979584
2016-01-29 20:35:53     Dump the side-table for tag: 1 with group count: 1 into file: file:/tmp/hadoop/77783d75-f81f-42c4-90c2-1193e7bdb4b2/hive_2016-01-29_20-35-07_662_928852317845709878-1/-local-10015/HashTable-Stage-14/MapJoin-mapfile31--.hashtable
2016-01-29 20:35:53     Uploaded 1 File to: file:/tmp/hadoop/77783d75-f81f-42c4-90c2-1193e7bdb4b2/hive_2016-01-29_20-35-07_662_928852317845709878-1/-local-10015/HashTable-Stage-14/MapJoin-mapfile31--.hashtable (294 bytes)
2016-01-29 20:35:53     End of local task; Time Taken: 1.221 sec.
Execution completed successfully
MapredLocal task succeeded
Launching Job 3 out of 9
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1454117877134_0011, Tracking URL = http://hadoop:8088/proxy/application_1454117877134_0011/
Kill Command = /usr/local/hadoop/bin/hadoop job  -kill job_1454117877134_0011
Hadoop job information for Stage-14: number of mappers: 1; number of reducers: 0
2016-01-29 20:36:03,853 Stage-14 map = 0%,  reduce = 0%
2016-01-29 20:36:15,685 Stage-14 map = 100%,  reduce = 0%, Cumulative CPU 4.52 sec
MapReduce Total cumulative CPU time: 4 seconds 520 msec
Ended Job = job_1454117877134_0011
```

```
Stage-18 is filtered out by condition resolver.
Stage-19 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Execution log at: /tmp/hadoop/hadoop_20160129203507_879eb87a-16c8-4c13-9457-6f354efffc58.log
2016-01-29 20:36:26     Starting to launch local task to process map join;      maximum memory = 518979584
2016-01-29 20:36:27     Dump the side-table for tag: 0 with group count: 352 into file: file:/tmp/hadoop/77783d75-f81f-42c4-90c2-1193e7bdb4b2/hive_2016-01-29_20-35-07_662_928852317845709878-1/-local-10013/HashTable-Stage-12/MapJoin-mapfile20--.hashtable
2016-01-29 20:36:27     Uploaded 1 File to: file:/tmp/hadoop/77783d75-f81f-42c4-90c2-1193e7bdb4b2/hive_2016-01-29_20-35-07_662_928852317845709878-1/-local-10013/HashTable-Stage-12/MapJoin-mapfile20--.hashtable (35206 bytes)
2016-01-29 20:36:27     End of local task; Time Taken: 1.247 sec.
Execution completed successfully
MapredLocal task succeeded
Launching Job 5 out of 9
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1454117877134_0012, Tracking URL = http://hadoop:8088/proxy/application_1454117877134_0012/
Kill Command = /usr/local/hadoop/bin/hadoop job  -kill job_1454117877134_0012
Hadoop job information for Stage-12: number of mappers: 1; number of reducers: 0
2016-01-29 20:36:38,112 Stage-12 map = 0%,  reduce = 0%
2016-01-29 20:36:49,887 Stage-12 map = 100%,  reduce = 0%, Cumulative CPU 3.62 sec
MapReduce Total cumulative CPU time: 3 seconds 620 msec
Ended Job = job_1454117877134_0012
Execution log at: /tmp/hadoop/hadoop_20160129203507_879eb87a-16c8-4c13-9457-6f354efffc58.log
2016-01-29 20:36:58     Starting to launch local task to process map join;      maximum memory = 518979584
2016-01-29 20:36:59     Dump the side-table for tag: 1 with group count: 558 into file: file:/tmp/hadoop/77783d75-f81f-42c4-90c2-1193e7bdb4b2/hive_2016-01-29_20-35-07_662_928852317845709878-1/-local-10009/HashTable-Stage-5/MapJoin-mapfile01--.hashtable
2016-01-29 20:36:59     Uploaded 1 File to: file:/tmp/hadoop/77783d75-f81f-42c4-90c2-1193e7bdb4b2/hive_2016-01-29_20-35-07_662_928852317845709878-1/-local-10009/HashTable-Stage-5/MapJoin-mapfile01--.hashtable (36975 bytes)
2016-01-29 20:36:59     End of local task; Time Taken: 1.709 sec.
Execution completed successfully
MapredLocal task succeeded
Launching Job 6 out of 9
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1454117877134_0013, Tracking URL = http://hadoop:8088/proxy/application_1454117877134_0013/
Kill Command = /usr/local/hadoop/bin/hadoop job  -kill job_1454117877134_0013
Hadoop job information for Stage-5: number of mappers: 1; number of reducers: 1
2016-01-29 20:37:11,788 Stage-5 map = 0%,  reduce = 0%
2016-01-29 20:37:20,368 Stage-5 map = 100%,  reduce = 0%, Cumulative CPU 1.64 sec
2016-01-29 20:37:28,981 Stage-5 map = 100%,  reduce = 100%, Cumulative CPU 3.14 sec
MapReduce Total cumulative CPU time: 3 seconds 140 msec
Ended Job = job_1454117877134_0013
```

```
Launching Job 7 out of 9
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1454117877134_0014, Tracking URL = http://hadoop:8088/proxy/application_1454117877134_0014/
Kill Command = /usr/local/hadoop/bin/hadoop job  -kill job_1454117877134_0014
Hadoop job information for Stage-6: number of mappers: 1; number of reducers: 1
2016-01-29 20:37:43,485 Stage-6 map = 0%,  reduce = 0%
2016-01-29 20:37:52,056 Stage-6 map = 100%,  reduce = 0%, Cumulative CPU 1.11 sec
2016-01-29 20:38:00,654 Stage-6 map = 100%,  reduce = 100%, Cumulative CPU 2.53 sec
MapReduce Total cumulative CPU time: 2 seconds 530 msec
Ended Job = job_1454117877134_0014
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 3.18 sec   HDFS Read: 14294 HDFS Write: 123 SUCCESS
Stage-Stage-14: Map: 1   Cumulative CPU: 4.52 sec   HDFS Read: 95612669 HDFS Write: 34842 SUCCESS
Stage-Stage-12: Map: 1   Cumulative CPU: 3.62 sec   HDFS Read: 53734672 HDFS Write: 5012 SUCCESS
Stage-Stage-5: Map: 1  Reduce: 1   Cumulative CPU: 3.14 sec   HDFS Read: 20221 HDFS Write: 1172 SUCCESS
Stage-Stage-6: Map: 1  Reduce: 1   Cumulative CPU: 2.53 sec   HDFS Read: 7069 HDFS Write: 581 SUCCESS
Total MapReduce CPU Time Spent: 16 seconds 990 msec
OK
H8578   BERKSHIRE       HNE Medicare Advantage Plans    HNE Medicare Basic No Rx (HMO)  123.9   4 out of 5 stars        4.5 out of 5 stars
H8578   BERKSHIRE       HNE Medicare Advantage Plans    HNE Medicare Value (HMO)        124.9   4 out of 5 stars        4.5 out of 5 stars
R7444   BERKSHIRE       UnitedHealthcare        AARP MedicareComplete Choice (Regional PPO)     144.9   4 out of 5 stars        3.5 out of 5 stars
H8578   BERKSHIRE       HNE Medicare Advantage Plans    HNE Medicare Basic (HMO)        179.9   4 out of 5 stars        4.5 out of 5 stars
H8578   BERKSHIRE       HNE Medicare Advantage Plans    HNE Medicare Premium No Rx (HMO)        193.9   4 out of 5 stars        4.5 out of 5 stars
Time taken: 175.28 seconds, Fetched: 5 row(s)
```

Screen Shot -4: Without subqueries/temporary tables and with Bucket Clustering

```
hive> SELECT contract_id,countyname,org_name, plan_name, max(premcalc(benefit)) prem,star_rating_Current, star_rating_previous
   > FROM countybuckettable
   > join countynamestable ON (countynamestable.FIPScode = countybuckettable.countyFIPcode)
   > JOIN planservicetable on (planservicetable.planid = countybuckettable.plan_id and countybuckettable.contract_id = planservicetable.contractid)
   > left outer join starratetable ON (countybuckettable.contract_id  = starratetable.ContractId)
```

```
   > where countybuckettable.countyFIPcode = "25003"
   > AND CategoryCode="1" AND (instr(benefit,"monthly premium")>0 OR instr(benefit,"per month")>0  OR instr(benefit,"per year")>0)
   > AND summary_score = 'Overall Star Rating'
   > Group by contract_id,countyname,org_name, plan_name,star_rating_Current, star_rating_previous
   > order by prem
   > LIMIT 5;
Query ID = hadoop_20160129203955_db311880-4070-489c-88e2-550ddb59b324
Total jobs = 6
Execution log at: /tmp/hadoop/hadoop_20160129203955_db311880-4070-489c-88e2-550ddb59b324.log
2016-01-29 20:40:01     Starting to launch local task to process map join;      maximum memory = 518979584
2016-01-29 20:40:03           Dump the side-table for tag: 1 with group count: 1 into file: file:/tmp/hadoop/77783d75-f81f-42c4-90c2-1193e7bdb4b2/hive_2016-01-29_20-39-55_162_8916445229526518610-1/-local-10014/HashTable-Stage-13/MapJoin-mapfile81--.hashtable
2016-01-29 20:40:03           Uploaded 1 File to: file:/tmp/hadoop/77783d75-f81f-42c4-90c2-1193e7bdb4b2/hive_2016-01-29_20-39-55_162_8916445229526518610-1/-local-10014/HashTable-Stage-13/MapJoin-mapfile81--.hashtable (867 bytes)
2016-01-29 20:40:03     End of local task; Time Taken: 2.024 sec.
Execution completed successfully
MapredLocal task succeeded
Launching Job 1 out of 6
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1454117877134_0015, Tracking URL = http://hadoop:8088/proxy/application_1454117877134_0015/
Kill Command = /usr/local/hadoop/bin/hadoop job  -kill job_1454117877134_0015
Hadoop job information for Stage-13: number of mappers: 1; number of reducers: 0
2016-01-29 20:40:15,779 Stage-13 map = 0%,  reduce = 0%
2016-01-29 20:40:28,577 Stage-13 map = 100%,  reduce = 0%, Cumulative CPU 5.2 sec
MapReduce Total cumulative CPU time: 5 seconds 200 msec
Ended Job = job_1454117877134_0015
Stage-15 is filtered out by condition resolver.
Stage-16 is selected by condition resolver.
Stage-2 is filtered out by condition resolver.
Execution log at: /tmp/hadoop/hadoop_20160129203955_db311880-4070-489c-88e2-550ddb59b324.log
2016-01-29 20:40:38     Starting to launch local task to process map join;      maximum memory = 518979584
2016-01-29 20:40:40           Dump the side-table for tag: 0 with group count: 352 into file: file:/tmp/hadoop/77783d75-f81f-42c4-90c2-1193e7bdb4b2/hive_2016-01-29_20-39-55_162_8916445229526518610-1/-local-10012/HashTable-Stage-11/MapJoin-mapfile70--.hashtable
2016-01-29 20:40:41           Uploaded 1 File to: file:/tmp/hadoop/77783d75-f81f-42c4-90c2-1193e7bdb4b2/hive_2016-01-29_20-39-55_162_8916445229526518610-1/-local-10012/HashTable-Stage-11/MapJoin-mapfile70--.hashtable (1054074 bytes)
2016-01-29 20:40:41     End of local task; Time Taken: 2.409 sec.
```

Execution completed successfully

MapredLocal task succeeded

Launching Job 3 out of 6

Number of reduce tasks is set to 0 since there's no reduce operator

Starting Job = job_1454117877134_0016, Tracking URL = http://hadoop:8088/proxy/application_1454117877134_0016/

Kill Command = /usr/local/hadoop/bin/hadoop job -kill job_1454117877134_0016

Hadoop job information for Stage-11: number of mappers: 1; number of reducers: 0

2016-01-29 20:40:51,424 Stage-11 map = 0%, reduce = 0%

2016-01-29 20:41:02,177 Stage-11 map = 100%, reduce = 0%, Cumulative CPU 4.06 sec

MapReduce Total cumulative CPU time: 4 seconds 60 msec

Ended Job = job_1454117877134_0016

Execution log at: /tmp/hadoop/hadoop_20160129203955_db311880-4070-489c-88e2-550ddb59b324.log

2016-01-29 20:41:10 Starting to launch local task to process map join; maximum memory = 518979584

2016-01-29 20:41:11 Dump the side-table for tag: 1 with group count: 642 into file: file:/tmp/hadoop/77783d75-f81f-42c4-90c2-1193e7bdb4b2/hive_2016-01-29_20-39-55_162_8916445229526518610-1/-local-10008/HashTable-Stage-4/MapJoin-mapfile51--.hashtable

2016-01-29 20:41:11 Uploaded 1 File to: file:/tmp/hadoop/77783d75-f81f-42c4-90c2-1193e7bdb4b2/hive_2016-01-29_20-39-55_162_8916445229526518610-1/-local-10008/HashTable-Stage-4/MapJoin-mapfile51--.hashtable (157660 bytes)

2016-01-29 20:41:11 End of local task; Time Taken: 1.626 sec.

Execution completed successfully

MapredLocal task succeeded

Launching Job 4 out of 6

Number of reduce tasks not specified. Estimated from input data size: 1

In order to change the average load for a reducer (in bytes):

set hive.exec.reducers.bytes.per.reducer=<number>

In order to limit the maximum number of reducers:

set hive.exec.reducers.max=<number>

In order to set a constant number of reducers:

set mapreduce.job.reduces=<number>

Starting Job = job_1454117877134_0017, Tracking URL = http://hadoop:8088/proxy/application_1454117877134_0017/

Kill Command = /usr/local/hadoop/bin/hadoop job -kill job_1454117877134_0017

Hadoop job information for Stage-4: number of mappers: 1; number of reducers: 1

2016-01-29 20:41:23,595 Stage-4 map = 0%, reduce = 0%

2016-01-29 20:41:34,482 Stage-4 map = 100%, reduce = 0%, Cumulative CPU 3.48 sec

2016-01-29 20:41:45,211 Stage-4 map = 100%, reduce = 100%, Cumulative CPU 5.09 sec

MapReduce Total cumulative CPU time: 5 seconds 90 msec

```
Ended Job = job_1454117877134_0017

Launching Job 5 out of 6

Number of reduce tasks determined at compile time: 1

In order to change the average load for a reducer (in bytes):

  set hive.exec.reducers.bytes.per.reducer=<number>

In order to limit the maximum number of reducers:

  set hive.exec.reducers.max=<number>

In order to set a constant number of reducers:

  set mapreduce.job.reduces=<number>

Starting Job = job_1454117877134_0018, Tracking URL = http://hadoop:8088/proxy/application_1454117877134_0018/

Kill Command = /usr/local/hadoop/bin/hadoop job  -kill job_1454117877134_0018

Hadoop job information for Stage-5: number of mappers: 1; number of reducers: 1

2016-01-29 20:42:00,312 Stage-5 map = 0%,  reduce = 0%

2016-01-29 20:42:08,896 Stage-5 map = 100%,  reduce = 0%, Cumulative CPU 1.08 sec

2016-01-29 20:42:18,536 Stage-5 map = 100%,  reduce = 100%, Cumulative CPU 2.71 sec

MapReduce Total cumulative CPU time: 2 seconds 710 msec

Ended Job = job_1454117877134_0018

MapReduce Jobs Launched:

Stage-Stage-13: Map: 1   Cumulative CPU: 5.2 sec   HDFS Read: 95613517 HDFS Write: 1355810 SUCCESS

Stage-Stage-11: Map: 1   Cumulative CPU: 4.06 sec   HDFS Read: 53734321 HDFS Write: 192000 SUCCESS

Stage-Stage-4: Map: 1  Reduce: 1   Cumulative CPU: 5.09 sec   HDFS Read: 208987 HDFS Write: 1172 SUCCESS

Stage-Stage-5: Map: 1  Reduce: 1   Cumulative CPU: 2.71 sec   HDFS Read: 7075 HDFS Write: 581 SUCCESS

Total MapReduce CPU Time Spent: 17 seconds 60 msec

OK

H8578   BERKSHIRE       HNE Medicare Advantage Plans    HNE Medicare Basic No Rx (HMO)  123.9   4 out of 5 stars        4.5 out of 5 stars

H8578   BERKSHIRE       HNE Medicare Advantage Plans    HNE Medicare Value (HMO)        124.9   4 out of 5 stars        4.5 out of 5 stars

R7444   BERKSHIRE       UnitedHealthcare        AARP MedicareComplete Choice (Regional PPO)     144.9   4 out of 5 stars        3.5 out of 5 stars

H8578   BERKSHIRE       HNE Medicare Advantage Plans    HNE Medicare Basic (HMO)        179.9   4 out of 5 stars        4.5 out of 5 stars

H8578   BERKSHIRE       HNE Medicare Advantage Plans    HNE Medicare Premium No Rx (HMO)        193.9   4 out of 5 stars        4.5 out of 5 stars

Time taken: 145.938 seconds, Fetched: 5 row(s)
```

Screen Shot -5: Sentiment analysis using twitter messages

```
SELECT  Planprovider, presence FROM  tweetpresence ORDER BYpresence desc limit 5;
FAILED: ParseException line 1:57 missing BY at 'BYpresence' near '<EOF>'
hive> SELECT  Planprovider, presence FROM  tweetpresence ORDER BY presence desc limit 5;
Query ID = hadoop_20160129212137_42b24ac6-5a67-4e62-be0e-e947380e21c8
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1454117877134_0023, Tracking URL = http://hadoop:8088/proxy/application_1454117877134_0023/
Kill Command = /usr/local/hadoop/bin/hadoop job  -kill job_1454117877134_0023
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2016-01-29 21:21:54,099 Stage-1 map = 0%,  reduce = 0%
2016-01-29 21:22:17,549 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 5.11 sec
2016-01-29 21:22:32,990 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 7.39 sec
MapReduce Total cumulative CPU time: 7 seconds 390 msec
Ended Job = job_1454117877134_0023
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 7.39 sec   HDFS Read: 6297 HDFS Write: 70 SUCCESS
Total MapReduce CPU Time Spent: 7 seconds 390 msec
OK
HUMANA  840.0
Aetna   93.0
Westmoreland    78.0
MedStar 27.0
BlueCross       12.0
Time taken: 56.351 seconds, Fetched: 5 row(s)
```