Mahout algorithms & use cases

Algorithm #1: Decision Forest to classify data

```
$HADOOP_HOME/bin/hadoop jar $MAHOUT_HOME/examples/target/mahout-examples-<version>-job.jar
org.apache.mahout.classifier.df.mapreduce.TestForest -i nsl-kdd/KDDTest+.arff -ds nsl-kdd/KDDTrain+.info -m nsl-forest -a
-mr -o predictions
```

Use Case #1: Can be used to solve classification problems such as deciding whether to buy or not buy, whether to build or not etc.

Algorithm #2: Hidden Markov Models

```
$ echo "0 1 2 2 2 1 1 0 0 3 3 3 2 1 2 1 1 1 1 2 2 2 0 0 0 0 0 0 2 2 2 0 0 0 0 0 0 2 2 2 3 3 3 3 3 3 2 3 2 3 2 3 2 1 3 0 0 0 1 0 1 0 2 1
2 1 2 1 2 3 3 3 3 2 2 3 2 1 1 0" > hmm-input
$ export MAHOUT_LOCAL=true
$ $MAHOUT_HOME/bin/mahout baumwelch -i hmm-input -o hmm-model -nh 3 -no 4 -e .0001 -m 1000
$ $MAHOUT_HOME/bin/mahout hmmpredict -m hmm-model -o hmm-predictions -l 10
```

Use Case #2: Can be used for speech recognition, natural language processing etc.

Mahout algorithms & use cases

Algorithm #3: K-Means clustering

```
// run the CanopyDriver job
CanopyDriver.runJob("testdata", "output"
ManhattanDistanceMeasure.class.getName(), (float) 3.1, (float) 2.1, false);

// now run the KMeansDriver job
KMeansDriver.runJob("testdata", "output/clusters-0", "output",
EuclideanDistanceMeasure.class.getName(), "0.001", "10", true);

bin/mahout kmeans \
   -i <input vectors directory> \
   -c <input clusters directory> \
   -o <output working directory> \
   -k <optional number of initial clusters to sample from input vectors> \
   -dm <DistanceMeasure> \
   -x <maximum number of iterations> \
   -cd <optional convergence delta. Default is 0.5> \
   -ow <overwrite output directory if present>
   -cl <run input vector clustering after computing Canopies>
   -xm <execution method: sequential or mapreduce>
```

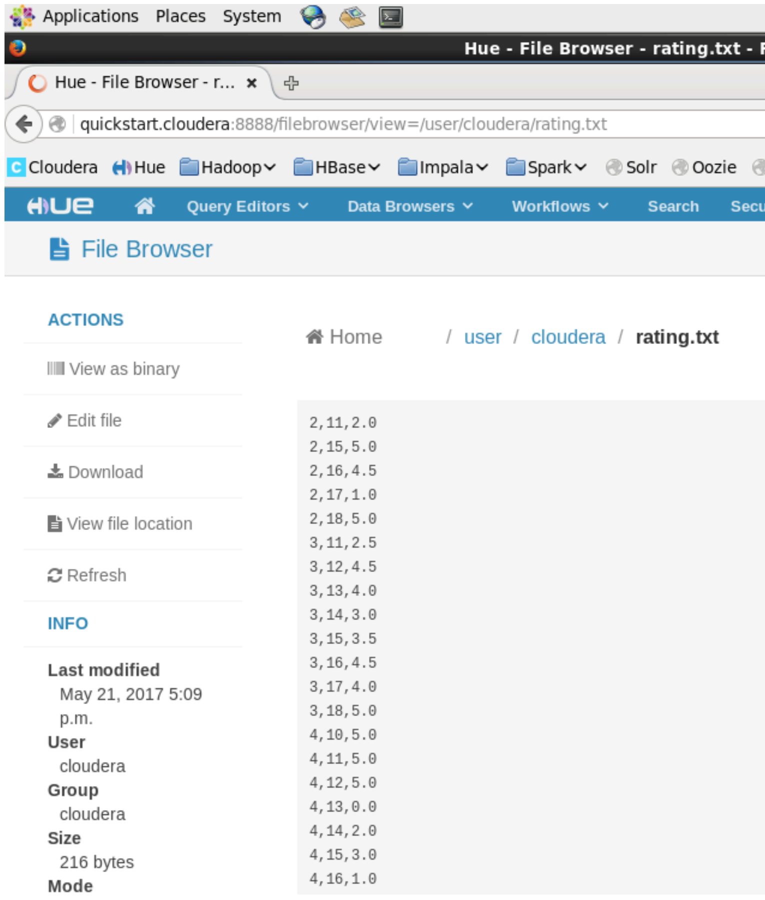
Use Case #3: Can be used to compute initial clusters for k-Kmeans in order to cluster data to find marketing segments, customer profile etc.

Build recommendation using Apache Mahout

Step 1:

Prepare and load data on Hadoop dataset in form user id, prod id, ratings:

-> The data was copied from the provided dataset and pasted in a file using the file browser interface.


Applications Places System
Hue - File Browser - rating.txt -
Hue - File Browser - r...
quickstart.cloudera:8888/filebrowser/view=/user/cloudera/rating.txt
Cloudera Hue Hadoop HBase Impala Spark Solr Oozie
Query Editors Data Browsers Workflows Search
File Browser
ACTIONS
View as binary
Edit file
Download
View file location
Refresh
INFO
Home / user / cloudera / rating.txt

Last modified
May 21, 2017 5:09 p.m.
User
cloudera
Group
cloudera
Size
216 bytes
Mode

```
2,11,2.0
2,15,5.0
2,16,4.5
2,17,1.0
2,18,5.0
3,11,2.5
3,12,4.5
3,13,4.0
3,14,3.0
3,15,3.5
3,16,4.5
3,17,4.0
3,18,5.0
4,10,5.0
4,11,5.0
4,12,5.0
4,13,0.0
4,14,2.0
4,15,3.0
4,16,1.0
```

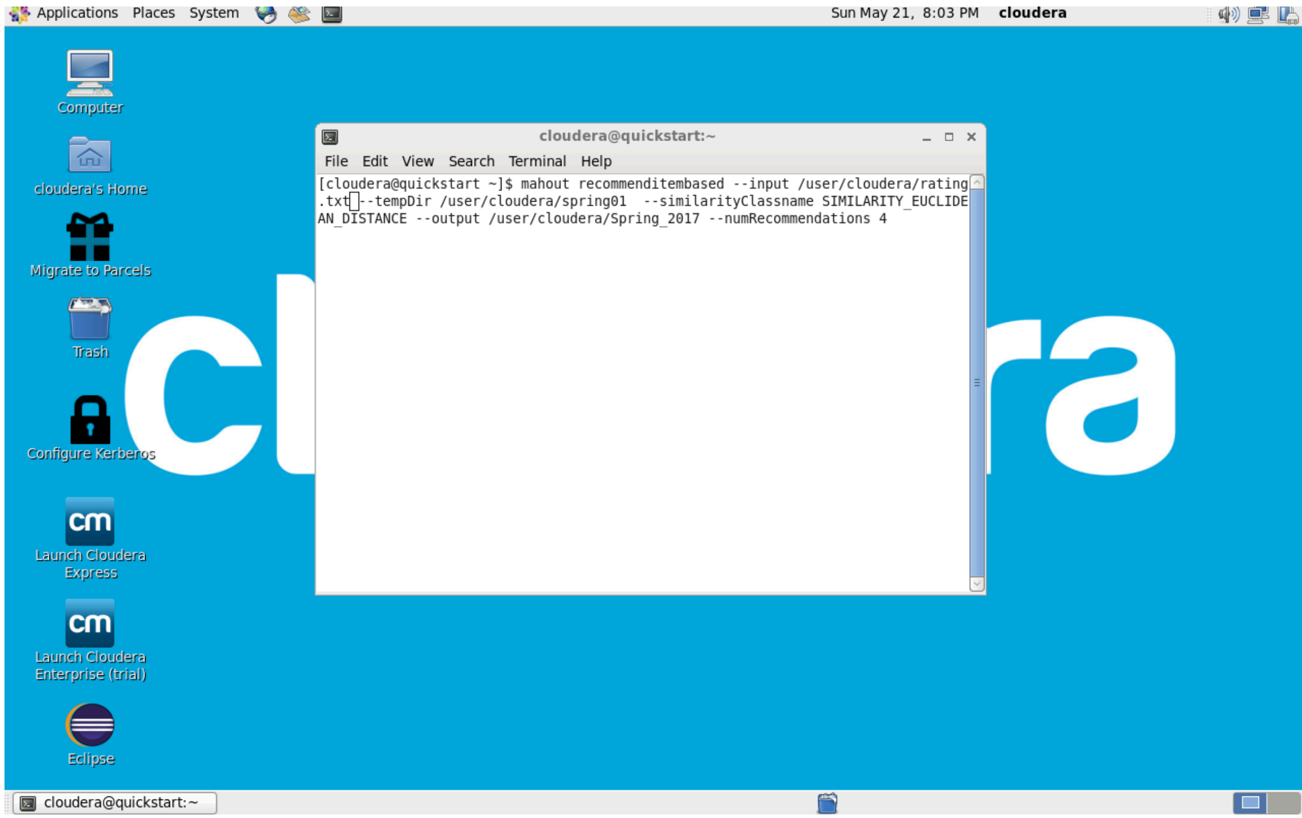
Build recommendation using Apache Mahout

Step 2:

Select and describe details of recommendation engine your are building

-> Below codes were used to create an item based recommendation engine in Apache Mahout. The item based recommendation engine was picked because of its versatility and ability to recommend the products based on the available data.

Codes used:
mahout recommenditembased --input /user/cloudera/recdemodata --tempDir /user/cloudera/sp1701 - similarityClassname SIMILARITY_EUCLIDEAN_DISTANCE --output /user/cloudera/sp_out_1701



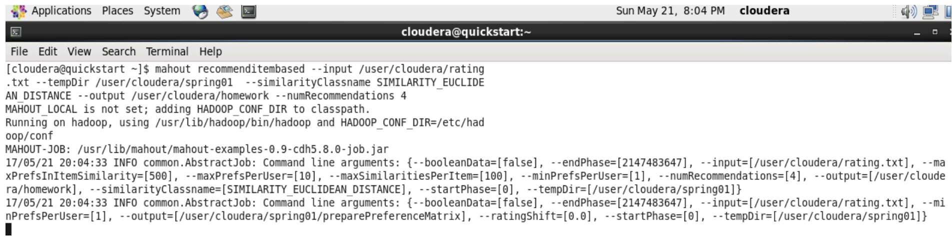
```
[cloudera@quickstart ~]$ mahout recommenditembased --input /user/cloudera/rating
.txt --tempDir /user/cloudera/spring01  --similarityClassname SIMILARITY_EUCLIDE
AN_DISTANCE --output /user/cloudera/Spring_2017 --numRecommendations 4
```

Build recommendation using Apache Mahout

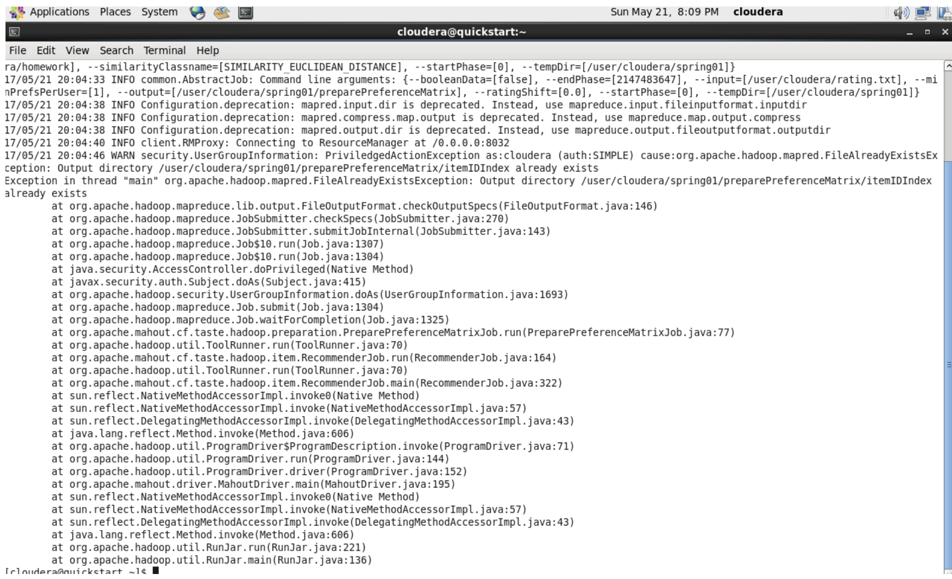
Step 3:

Run recommendation engine using Mahout:

-> Upon running the recommendation engine with the codes in previous slide, the following run-message showed up on the screen. The output are discussed in next slides.



```
[cloudera@quickstart ~]$ mahout recommenditembased --input /user/cloudera/rating
.txt --tempDir /user/cloudera/spring01  --similarityClassname SIMILARITY_EUCLIDE
AN_DISTANCE --output /user/cloudera/homework --numRecommendations 4
MAHOUT_LOCAL is not set; adding HADOOP_CONF_DIR to classpath.
Running on hadoop, using /usr/lib/hadoop/bin/hadoop and HADOOP_CONF_DIR=/etc/had
oop/conf
MAHOUT-JOB: /usr/lib/mahout/mahout-examples-0.9-cdh5.8.0-job.jar
17/05/21 20:04:33 INFO common.AbstractJob: Command line arguments: {--booleanData=[false], --endPhase=[2147483647], --input=[/user/cloudera/rating.txt], --ma
xPrefsInItemSimilarity=[500], --maxPrefsPerUser=[10], --maxSimilaritiesPerItem=[100], --minPrefsPerUser=[1], --numRecommendations=[4], --output=[/user/cloude
ra/homework], --similarityClassname=[SIMILARITY_EUCLIDEAN_DISTANCE], --startPhase=[0], --tempDir=[/user/cloudera/spring01]}
17/05/21 20:04:33 INFO common.AbstractJob: Command line arguments: {--booleanData=[false], --endPhase=[2147483647], --input=[/user/cloudera/rating.txt], --mi
nPrefsPerUser=[1], --output=[/user/cloudera/spring01/preparePreferenceMatrix], --ratingShift=[0.0], --startPhase=[0], --tempDir=[/user/cloudera/spring01]}
```

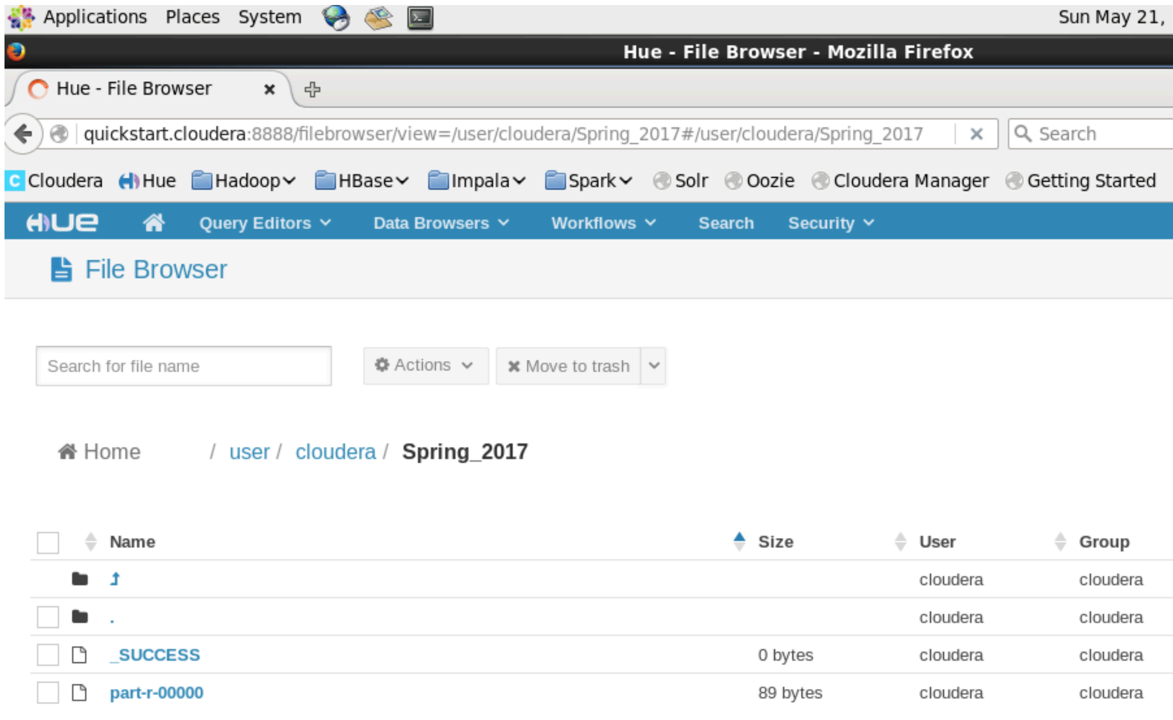


```
ra/homework], --similarityClassname=[SIMILARITY_EUCLIDEAN_DISTANCE], --startPhase=[0], --tempDir=[/user/cloudera/spring01]}
17/05/21 20:04:33 INFO common.AbstractJob: Command line arguments: {--booleanData=[false], --endPhase=[2147483647], --input=[/user/cloudera/rating.txt], --mi
nPrefsPerUser=[1], --output=[/user/cloudera/spring01/preparePreferenceMatrix], --ratingShift=[0.0], --startPhase=[0], --tempDir=[/user/cloudera/spring01]}
17/05/21 20:04:38 INFO Configuration.deprecation: mapred.input.dir is deprecated. Instead, use mapreduce.input.fileinputformat.inputdir
17/05/21 20:04:38 INFO Configuration.deprecation: mapred.compress.map.output is deprecated. Instead, use mapreduce.map.output.compress
17/05/21 20:04:38 INFO Configuration.deprecation: mapred.output.dir is deprecated. Instead, use mapreduce.output.fileoutputformat.outputdir
17/05/21 20:04:40 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
17/05/21 20:04:46 WARN security.UserGroupInformation: PriviledgedActionException as:cloudera (auth:SIMPLE) cause:org.apache.hadoop.mapred.FileAlreadyExistsEx
ception: Output directory /user/cloudera/spring01/preparePreferenceMatrix/itemIDIndex already exists
Exception in thread "main" org.apache.hadoop.mapred.FileAlreadyExistsException: Output directory /user/cloudera/spring01/preparePreferenceMatrix/itemIDIndex
already exists
        at org.apache.hadoop.mapreduce.lib.output.FileOutputFormat.checkOutputSpecs(FileOutputFormat.java:146)
        at org.apache.hadoop.mapreduce.JobSubmitter.checkSpecs(JobSubmitter.java:270)
        at org.apache.hadoop.mapreduce.JobSubmitter.submitJobInternal(JobSubmitter.java:143)
        at org.apache.hadoop.mapreduce.Job$10.run(Job.java:1307)
        at org.apache.hadoop.mapreduce.Job$10.run(Job.java:1304)
        at java.security.AccessController.doPrivileged(Native Method)
        at javax.security.auth.Subject.doAs(Subject.java:415)
        at org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInformation.java:1693)
        at org.apache.hadoop.mapreduce.Job.submit(Job.java:1304)
        at org.apache.hadoop.mapreduce.Job.waitForCompletion(Job.java:1325)
        at org.apache.mahout.cf.taste.hadoop.preparation.PreparePreferenceMatrixJob.run(PreparePreferenceMatrixJob.java:77)
        at org.apache.hadoop.util.ToolRunner.run(ToolRunner.java:70)
        at org.apache.mahout.cf.taste.hadoop.item.RecommenderJob.run(RecommenderJob.java:164)
        at org.apache.hadoop.util.ToolRunner.run(ToolRunner.java:70)
        at org.apache.mahout.cf.taste.hadoop.item.RecommenderJob.main(RecommenderJob.java:322)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:57)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
        at java.lang.reflect.Method.invoke(Method.java:606)
        at org.apache.hadoop.util.ProgramDriver$ProgramDescription.invoke(ProgramDriver.java:71)
        at org.apache.hadoop.util.ProgramDriver.run(ProgramDriver.java:144)
        at org.apache.hadoop.util.ProgramDriver.driver(ProgramDriver.java:152)
        at org.apache.mahout.driver.MahoutDriver.main(MahoutDriver.java:195)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:57)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
        at java.lang.reflect.Method.invoke(Method.java:606)
        at org.apache.hadoop.util.RunJar.run(RunJar.java:221)
        at org.apache.hadoop.util.RunJar.main(RunJar.java:136)
[cloudera@quickstart ~]$
```

Step 4:

Build recommendation using Apache Mahout

Output validation: Upon finishing the run, we were able to cross validate and check that the run was successful. The screenshot on the right shows the success message and output files.


Applications Places System
Sun May 21,
Hue - File Browser - Mozilla Firefox
Hue - File Browser
quickstart.cloudera:8888/filebrowser/view=/user/cloudera/Spring_2017#/user/cloudera/Spring_2017
Search
Cloudera Hue Hadoop HBase Impala Spark Solr Oozie Cloudera Manager Getting Started
HUE Query Editors Data Browsers Workflows Search Security
File Browser
Search for file name
Actions
Move to trash
Home / user / cloudera / Spring_2017

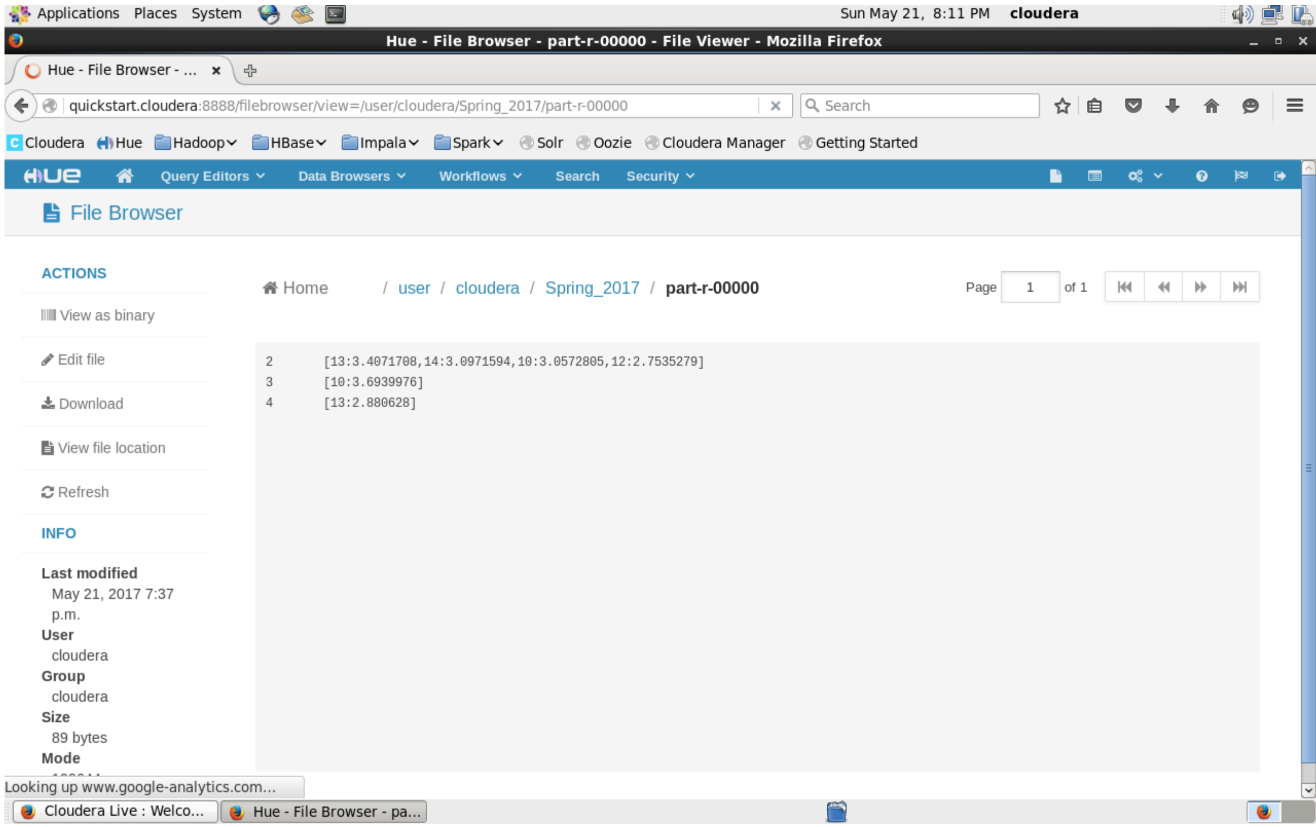
Name	Size	User	Group
↥		cloudera	cloudera
.		cloudera	cloudera
_SUCCESS	0 bytes	cloudera	cloudera
part-r-00000	89 bytes	cloudera	cloudera

Build recommendation using Apache Mahout

Step 4:

Output explanation:

The output on the right shows the three recommended products. The recommended products are 10 and 13 with their respective ratings immediately separated by comma.



Hue - File Browser - part-r-00000 - File Viewer - Mozilla Firefox

quickstart.cloudera:8888/filebrowser/view=/user/cloudera/Spring_2017/part-r-00000

File Browser

Home / user / cloudera / Spring_2017 / part-r-00000

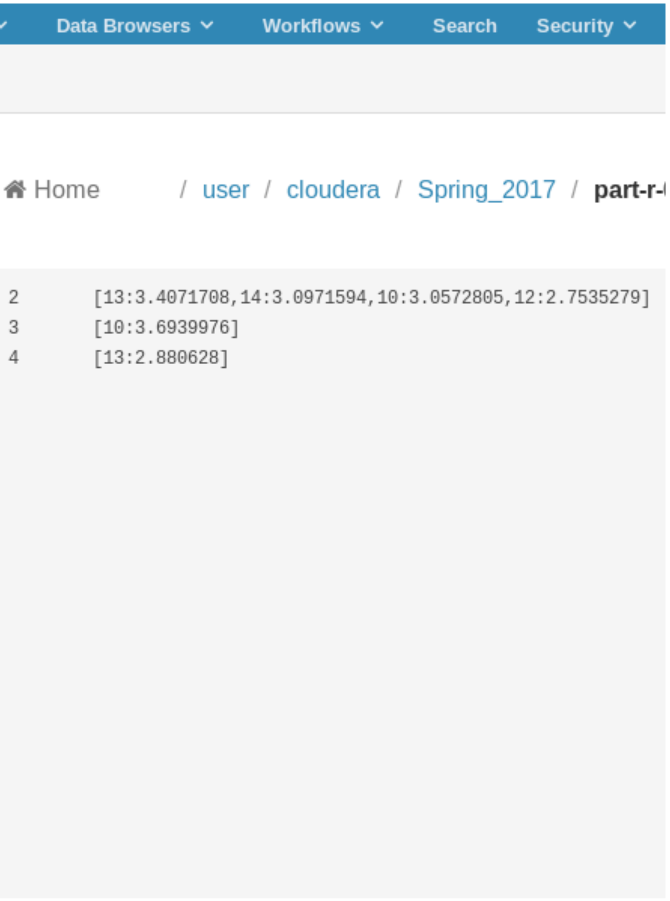
Page 1 of 1

```
2       [13:3.4071708,14:3.0971594,10:3.0572805,12:2.7535279]
3       [10:3.6939976]
4       [13:2.880628]
```

INFO

Last modified
May 21, 2017 7:37 p.m.
User
cloudera
Group
cloudera
Size
89 bytes
Mode

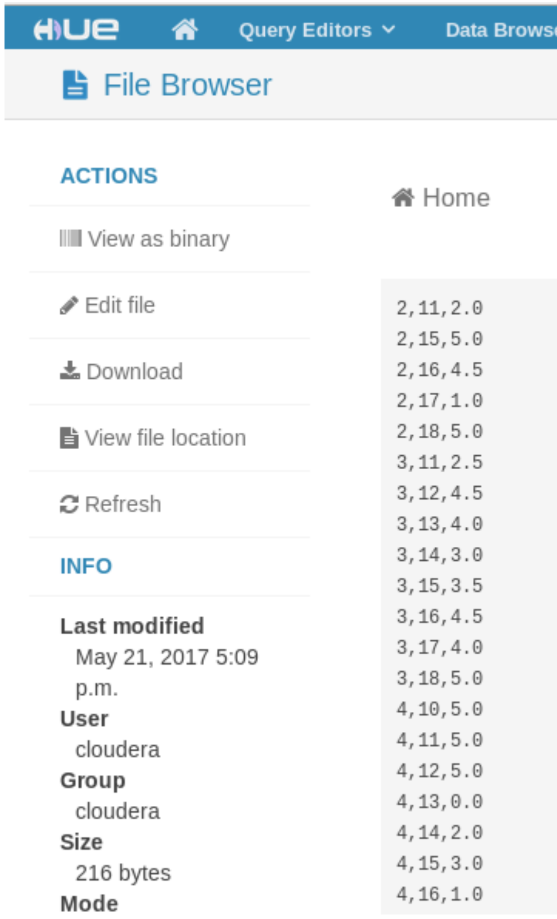
Build recommendation using Apache Mahout



```
2       [13:3.4071708,14:3.0971594,10:3.0572805,12:2.7535279]
3       [10:3.6939976]
4       [13:2.880628]
```

Comparison:

The screenshot on the right is the input file and the screenshot on the left is the output file. The input file has products from 11 thru 18 with different ratings. Based on the input file, we used Apache Mahout's item based recommendation engine to recommend three additional products with different ratings. The results on the left are the new products with their respective ratings that are recommended based on the data to the right.



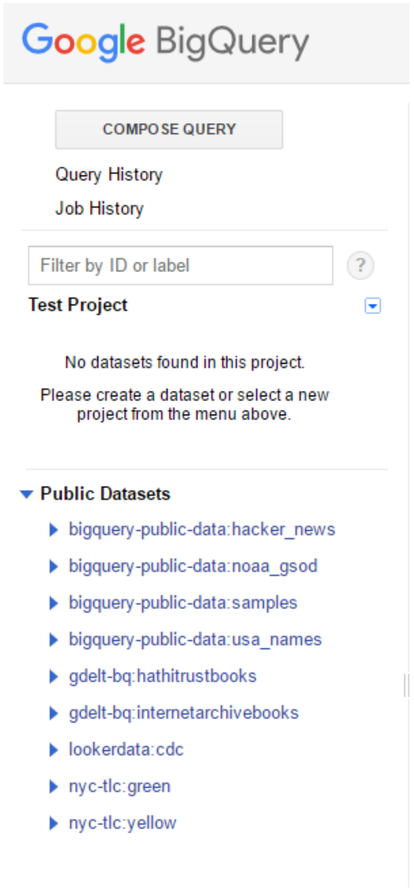
Last modified
May 21, 2017 5:09 p.m.
User
cloudera
Group
cloudera
Size
216 bytes
Mode

```
2,11,2.0
2,15,5.0
2,16,4.5
2,17,1.0
2,18,5.0
3,11,2.5
3,12,4.5
3,13,4.0
3,14,3.0
3,15,3.5
3,16,4.5
3,17,4.0
3,18,5.0
4,10,5.0
4,11,5.0
4,12,5.0
4,13,0.0
4,14,2.0
4,15,3.0
4,16,1.0
```

Analytics on Cloud using Google Big Query

Create Google Big Query account and show list of public tables, which table you liked most

-> We created a Google Big Query account and found the list of public tables screenshot on the right. I found the US census data most helpful amongst all of the data sets available.



No datasets found in this project.

Please create a dataset or select a new project from the menu above.

Public Datasets

- bigquery-public-data:hacker_news
- bigquery-public-data:noaa_gsod
- bigquery-public-data:samples
- bigquery-public-data:usa_names
- gdelt-bq:hathitrustbooks
- gdelt-bq:internetarchivebooks
- lookerdata:cdc
- nyc-tlc:green
- nyc-tlc:yellow

Analytics on Cloud using Google Big Query

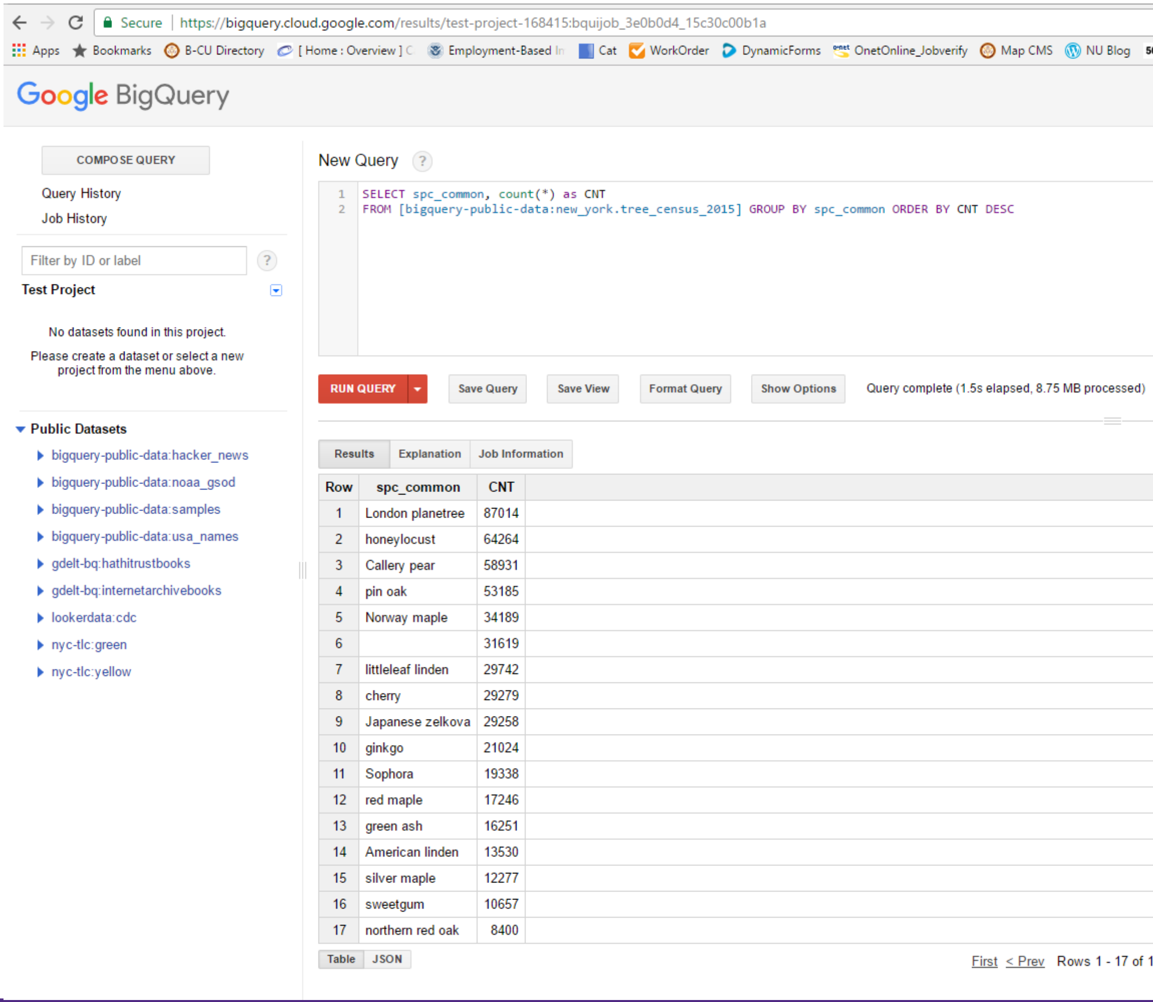
Run query and share results, explain what this query does?

Codes used:

SELECT spc_common, count(*) as CNT FROM [bigquery-public-data:new_york.tree_census_2015] GROUP BY spc_common ORDER BY CNT DESC

Result explanation:

The query resulted in the count of the number of line items in the new.york.tree.census_2015 dataset grouped by the type of spc_common. We then ordered the data in descending order based on the count of spc_common. It appears that London plantree, honeylocust and callery,pear were most common in 2015.


Secure | https://bigquery.cloud.google.com/results/test-project-168415:bquijob_3e0b0d4_15c30c00b1a
Apps Bookmarks B-CU Directory [Home : Overview] C Employment-Based In Cat WorkOrder DynamicForms OnetOnline_Jobverify Map CMS NU Blog
Google BigQuery
COMPOSE QUERY
Query History
Job History
Filter by ID or label
Test Project
No datasets found in this project.
Please create a dataset or select a new project from the menu above.
Public Datasets
bigquery-public-data:hacker_news
bigquery-public-data:noaa_gsod
bigquery-public-data:samples
bigquery-public-data:usa_names
gdelt-bq:hathitrustbooks
gdelt-bq:internetarchivebooks
lookerdata:cdc
nyc-tlc:green
nyc-tlc:yellow
New Query
RUN QUERY
Save Query
Save View
Format Query
Show Options
Results
Explanation
Job Information
Table
JSON
First
< Prev

```
SELECT spc_common, count(*) as CNT
FROM [bigquery-public-data:new_york.tree_census_2015] GROUP BY spc_common ORDER BY CNT DESC
```

Query complete (1.5s elapsed, 8.75 MB processed)

Row	spc_common	CNT
1	London planetree	87014
2	honeylocust	64264
3	Callery pear	58931
4	pin oak	53185
5	Norway maple	34189
6		31619
7	littleleaf linden	29742
8	cherry	29279
9	Japanese zelkova	29258
10	ginkgo	21024
11	Sophora	19338
12	red maple	17246
13	green ash	16251
14	American linden	13530
15	silver maple	12277
16	sweetgum	10657
17	northern red oak	8400

Rows 1 - 17 of 1

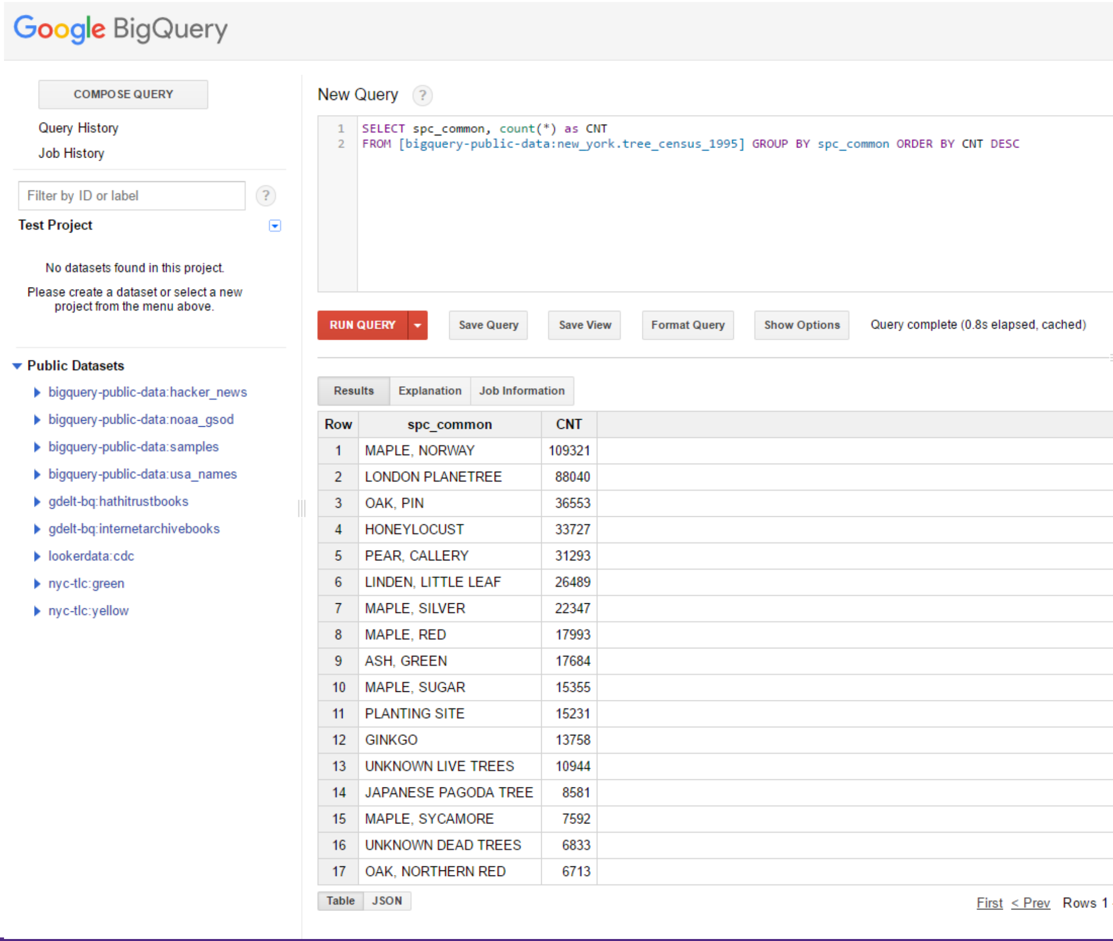
Analytics on Cloud using Google Big Query

Change query to run for 1995 Census and explain what changes you see since 1995 to 2015 in New york city.

Codes Used:
SELECT spc_common, count(*) as CNT
FROM [bigquery-public-data:new_york.tree_census_1995] GROUP BY spc_common ORDER BY CNT DESC

Results:

When changing the year in a similar data set from previous slide, the result was entirely different. Unlike 2015, it appears that Maple_Norway, London_Planetree and Oak,Pin had the highest census in 1995.



No datasets found in this project.

Please create a dataset or select a new project from the menu above.

```
SELECT spc_common, count(*) as CNT
FROM [bigquery-public-data:new_york.tree_census_1995] GROUP BY spc_common ORDER BY CNT DESC
```

Query complete (0.8s elapsed, cached)

Row	spc_common	CNT
1	MAPLE, NORWAY	109321
2	LONDON PLANETREE	88040
3	OAK, PIN	36553
4	HONEYLOCUST	33727
5	PEAR, CALLERY	31293
6	LINDEN, LITTLE LEAF	26489
7	MAPLE, SILVER	22347
8	MAPLE, RED	17993
9	ASH, GREEN	17684
10	MAPLE, SUGAR	15355
11	PLANTING SITE	15231
12	GINKGO	13758
13	UNKNOWN LIVE TREES	10944
14	JAPANESE PAGODA TREE	8581
15	MAPLE, SYCAMORE	7592
16	UNKNOWN DEAD TREES	6833
17	OAK, NORTHERN RED	6713

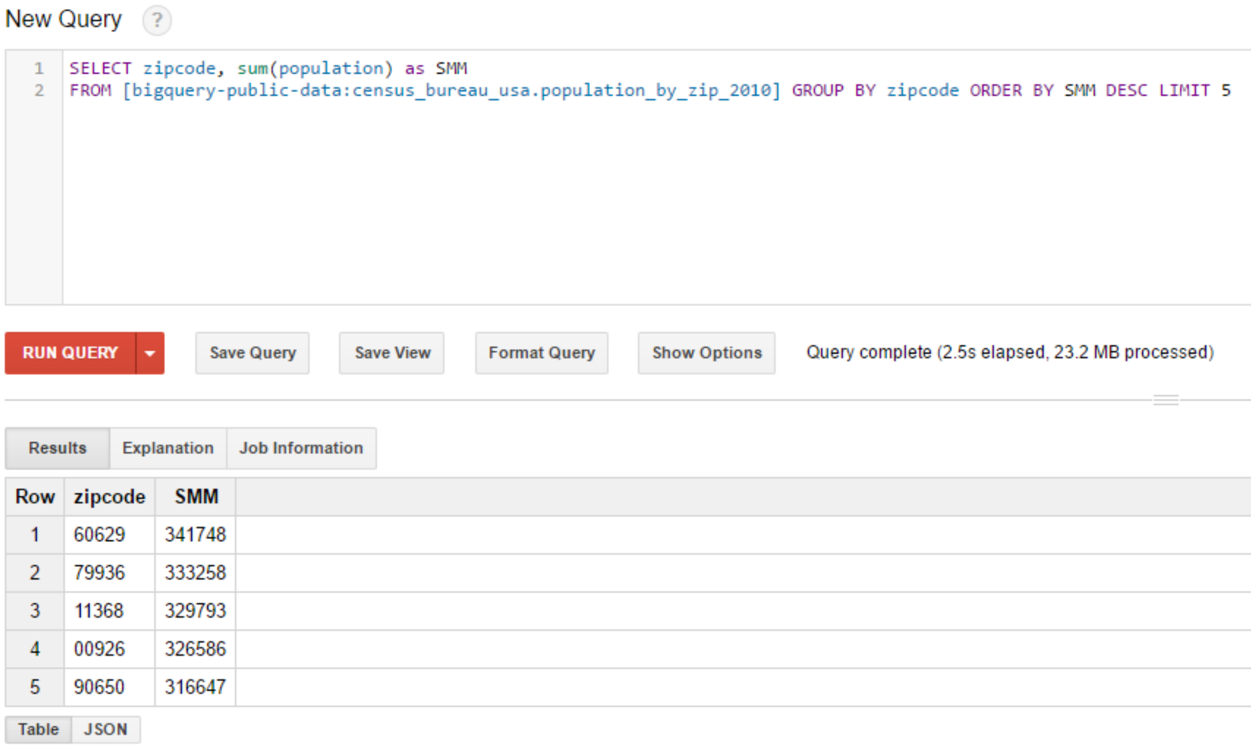
Analytics on Cloud using Google Big Query

BONUS Question #1

2. List top 5 zipcodes with highest population in USA USE table census bureau_usa in 2010

Codes Used:
SELECT zipcodes, sum(population) as SMM FROM [bigquery-public-data:census_bureau_usa.population_by_zip_2010] GROUP BY zipcode ORDER BY SMM DESC LIMIT 5

Results are presented on the screenshot to the right.



```
SELECT zipcode, sum(population) as SMM
FROM [bigquery-public-data:census_bureau_usa.population_by_zip_2010] GROUP BY zipcode ORDER BY SMM DESC LIMIT 5
```

Query complete (2.5s elapsed, 23.2 MB processed)

Row	zipcode	SMM
1	60629	341748
2	79936	333258
3	11368	329793
4	00926	326586
5	90650	316647

Northwestern

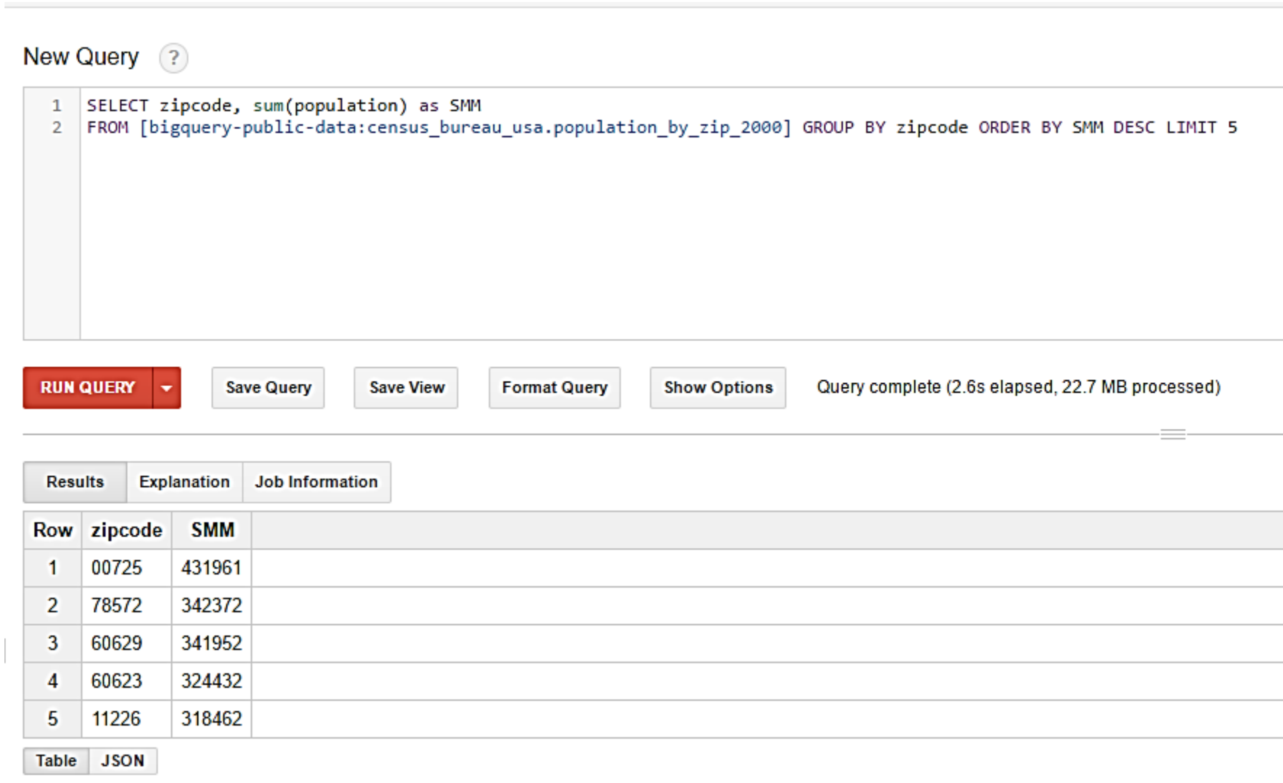
Analytics on Cloud using Google Big Query

BONUS Question #2

1. List top 5 zipcodes with highest population in USA USE table census_bureau_usa in 2000

Codes Used:
SELECT zipcodes, sum(population) as SMM FROM [bigquery-public-data:census_bureau_usa.population_by_zip_2000] GROUP BY zipcode ORDER BY SMM DESC LIMIT 5

Results are presented on the screenshot to the right.



```
SELECT zipcode, sum(population) as SMM
FROM [bigquery-public-data:census_bureau_usa.population_by_zip_2000] GROUP BY zipcode ORDER BY SMM DESC LIMIT 5
```

Query complete (2.6s elapsed, 22.7 MB processed)

Row	zipcode	SMM
1	00725	431961
2	78572	342372
3	60629	341952
4	60623	324432
5	11226	318462

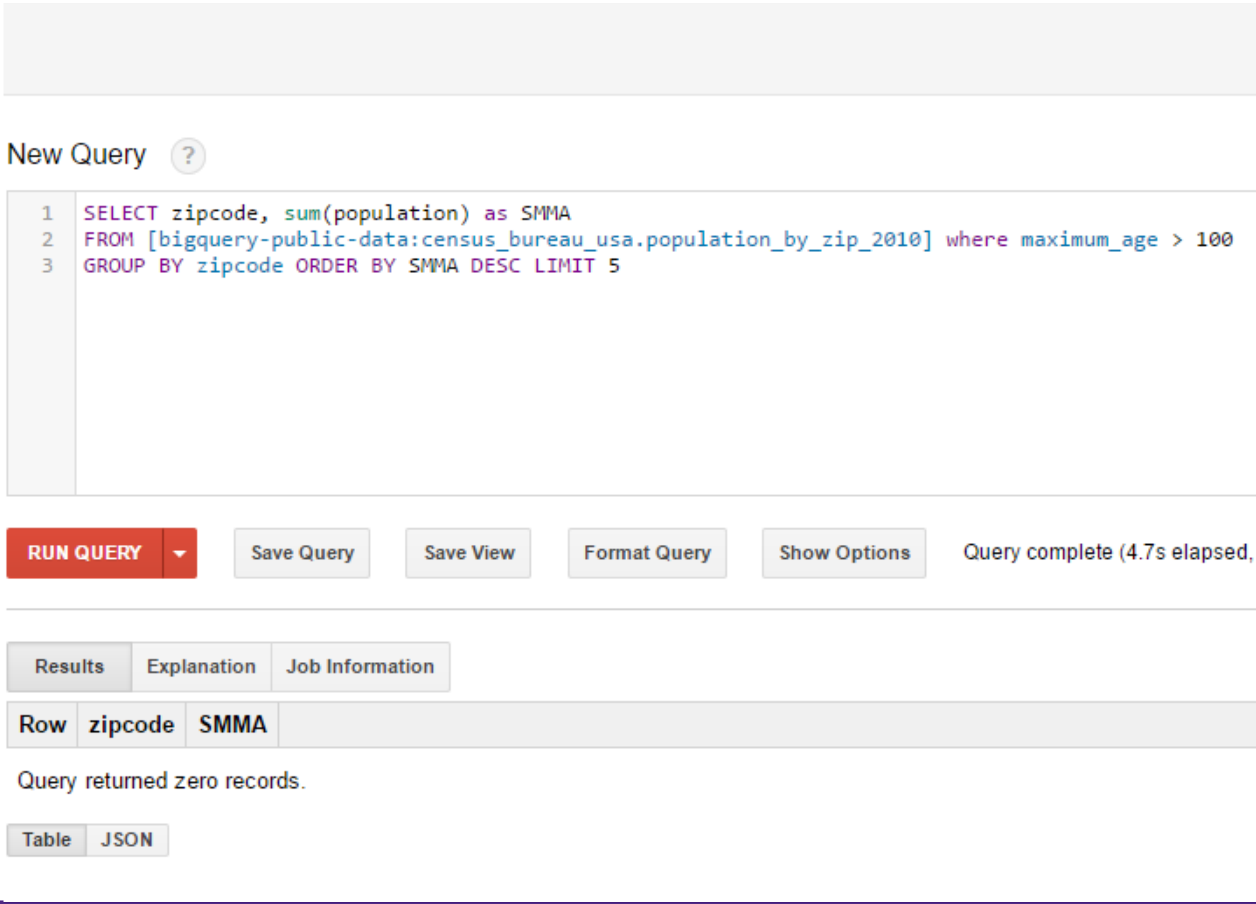
Analytics on Cloud using Google Big Query

BONUS Question #3

3. Per 2010 census which zipcode has more number of peoples with max age >100

Codes Used:
SELECT zipcodes, sum(population) as SMMA FROM [bigquery-public-data:census_bureau_usa.population_by_zip_2010] where maximum_age > 100 GROUP BY zipcode ORDER BY SMMA DESC LIMIT 5

Apparently, there were no entries in 2010 census data that has a zip code with number of peoples whose max age is greater than 100.



```
SELECT zipcode, sum(population) as SMMA
FROM [bigquery-public-data:census_bureau_usa.population_by_zip_2010] where maximum_age > 100
GROUP BY zipcode ORDER BY SMMA DESC LIMIT 5
```

Query complete (4.7s elapsed,

Row	zipcode	SMMA

Query returned zero records.

Thank You!