

MongoDB commands

1. Insert records into MongoDB Collection

Code:

```
db.cis436.insert(UserId: '9', ProductId: '500', Ratings:'5');
```

(Here, the insert command inserts new data in the collection.)

2. Update records into MongoDB collection

Code:

```
Db.cis436.update(  
  { UserId: "1" },  
  {  
    UserId: "2",  
    ProductId: 300,  
    Ratings:4  
  },  
  {upsert: true }  
)
```

(Here, the update command updates the collection with new data.)

MongoDB commands

3. Delete/Remove records from MongoDB collection

Code:

```
db.cis436.drop(UserId: '9')
```

(Here, the drop command drops data from the collection.)

4. Find/Search records from MongoDB Collection

Code:

```
db.cis436.find(UserId: '2')
```

(Here, the find command helps search the collection for the data one wants to search.)

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>
```

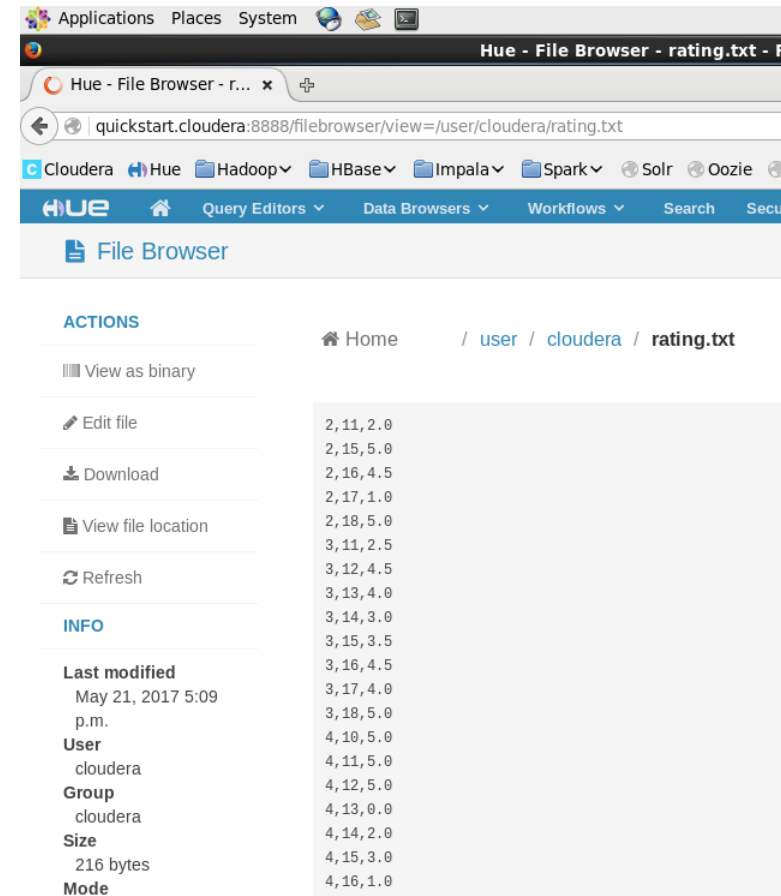
Use Case #3: Can be used to compute initial clusters for k-Kmeans in order to cluster data to find marketing segments, customer profile etc.

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.



Build recommendation using Apache Mahout

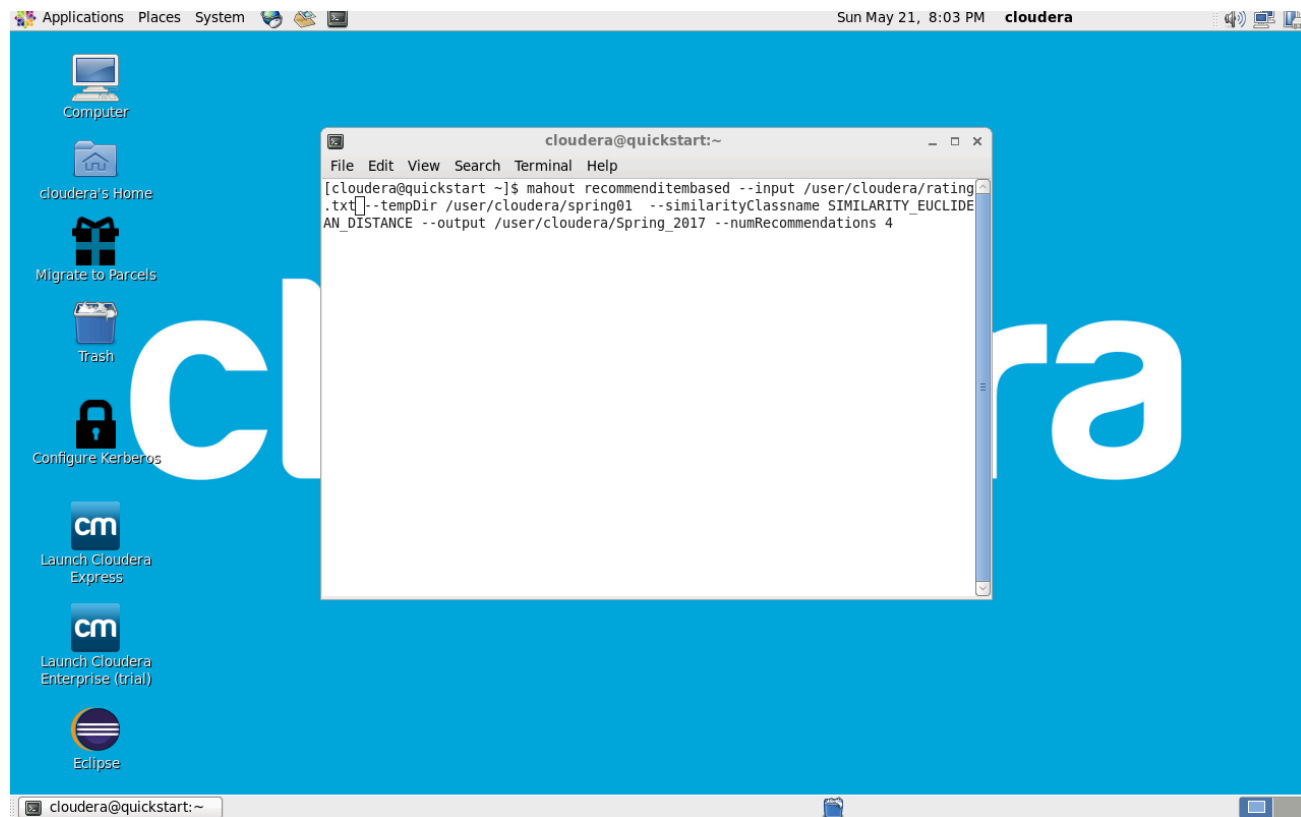
Step 2:

Select and describe details of recommendation engine you 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/recdmodata --tempDir /
user/cloudera/sp1701 - similarityClassname
SIMILARITY_EUCLIDEAN_DISTANCE --
output /user/cloudera/sp_out_1701
```

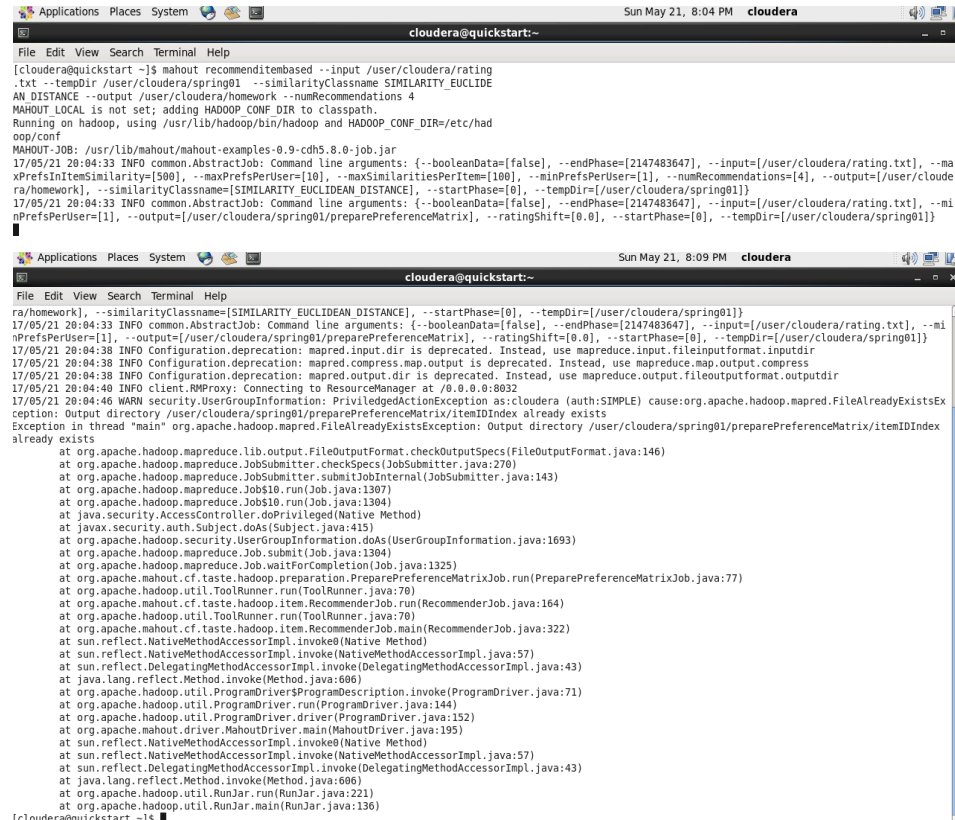


Build recommendation using Apache Mahout

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.

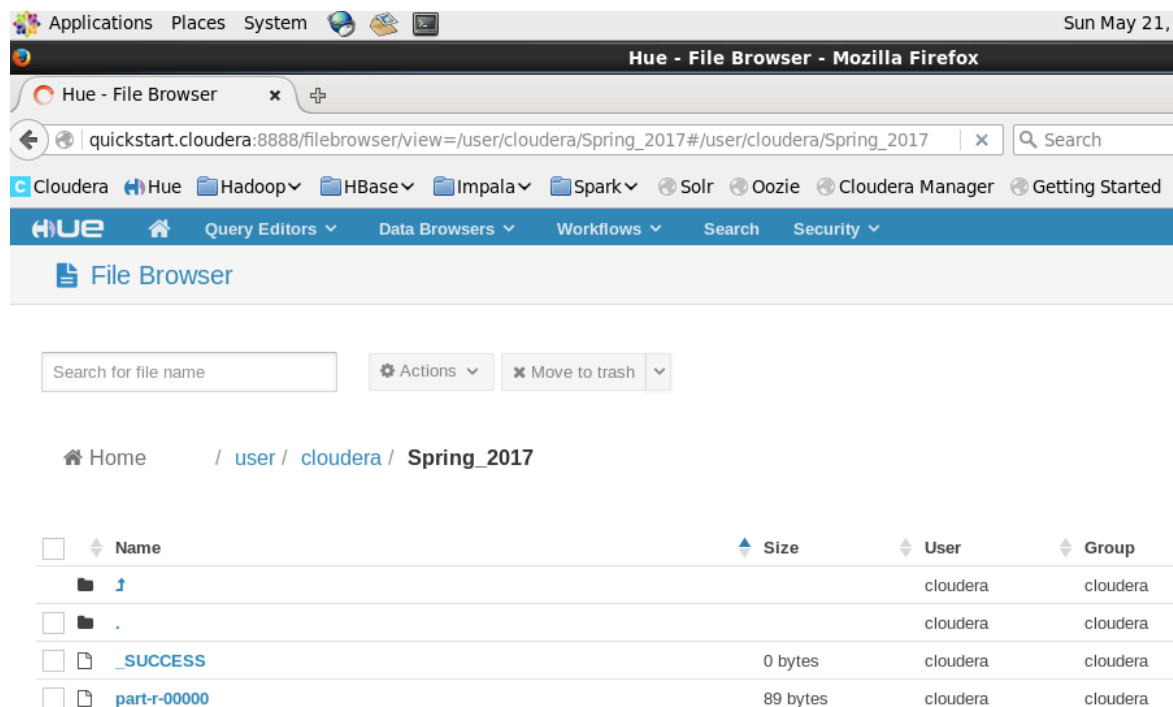


```
cloudera@quickstart:~  
File Edit View Search Terminal Help  
cloudera@quickstart:~$ mahout recommenditembased --input /user/cloudera/rating.txt --tempDir /user/cloudera/spring01 --similarityClassname SIMILARITY_EUCLIDEAN --output /user/cloudera/homework --numRecommendations 4  
MAHOUT_LOCAL is not set; adding HADOOP_CONF_DIR to classpath.  
Running on hadoop, using /usr/lib/hadoop/bin/hadoop and HADOOP_CONF_DIR=/etc/hadoop/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], --maxPrefsInItemSimilarity=[500], --maxPrefsPerUser=[10], --maxSimilaritiesPerItem=[100], --minPrefsPerUser=[1], --numRecommendations=[4], --output=[/user/cloudera/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], --minPrefsPerUser=[1], --output=[/user/cloudera/spring01/preparePreferenceMatrix], --ratingShift=[0.0], --startPhase=[0], --tempDir=[/user/cloudera/spring01]]  
  
cloudera@quickstart:~  
File Edit View Search Terminal Help  
17/05/21 20:04:33 INFO common.AbstractJob: Command line arguments: [--booleanData=[false], --endPhase=[2147483647], --input=[/user/cloudera/rating.txt], --minPrefsPerUser=[1], --output=[/user/cloudera/spring01/preparePreferenceMatrix], --ratingsShift=[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 20:04:40 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032  
17/05/21 20:04:46 WARN security.UserGroupInformation: PrivilegedActionException as:cloudera (auth:SIMPLE) cause:org.apache.hadoop.mapred.FileAlreadyExistsException: 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$18.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:~$
```

Build recommendation using Apache Mahout

Step 4:

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.

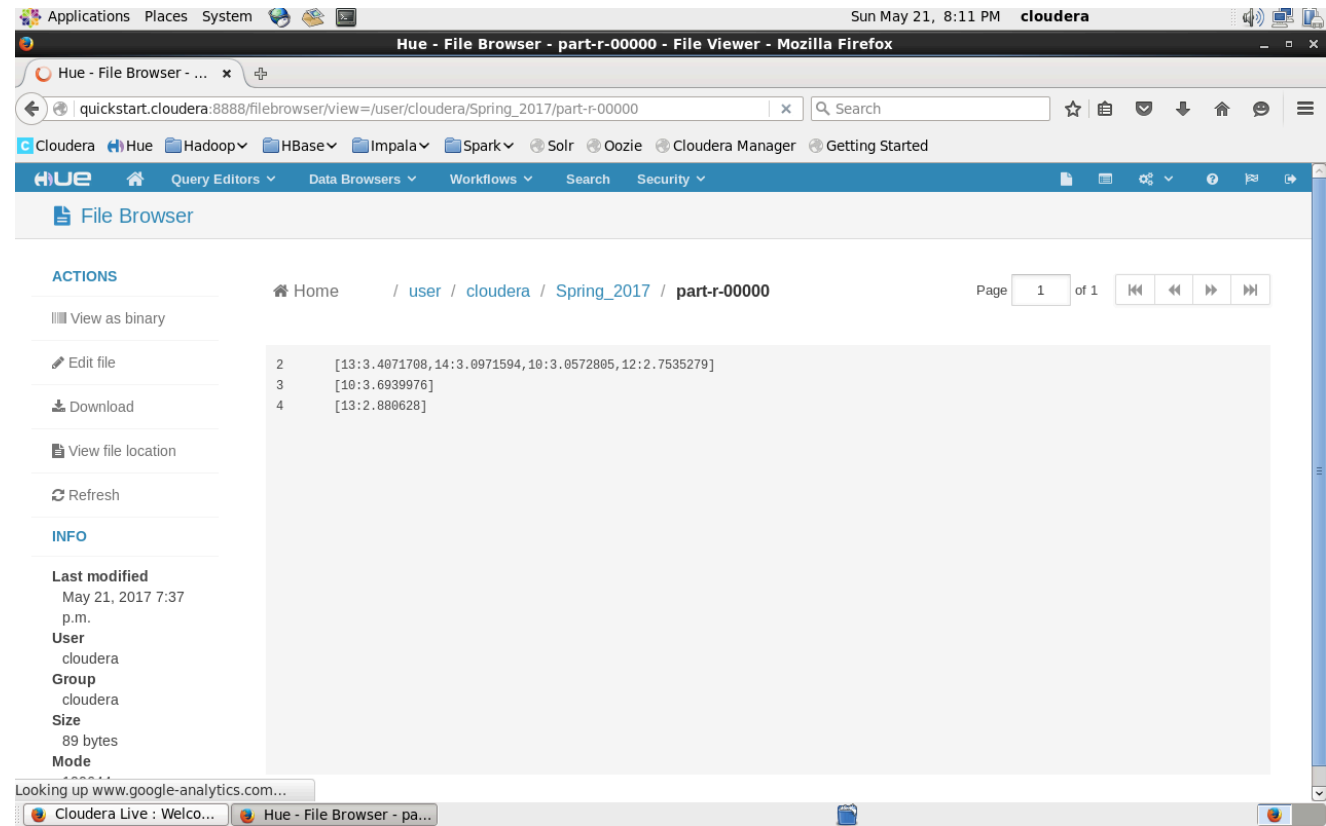


Build recommendation using Apache Mahout

Step 4:

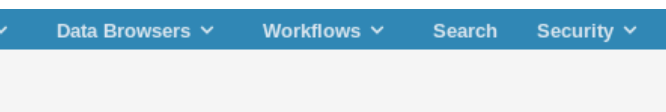
Output explanation:

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.



Build recommendation using Apache Mahout

Step 5:

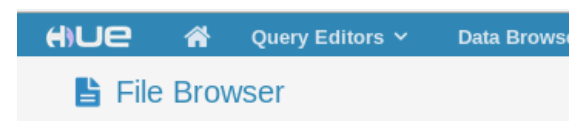


Home / user / cloudera / Spring_2017 / part-r

```
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.



ACTIONS

View as binary

Edit file

Download

View file location

Refresh

INFO

Last modified

May 21, 2017 5:09 p.m.

User

cloudera

Group

cloudera

Size

216 bytes

Mode

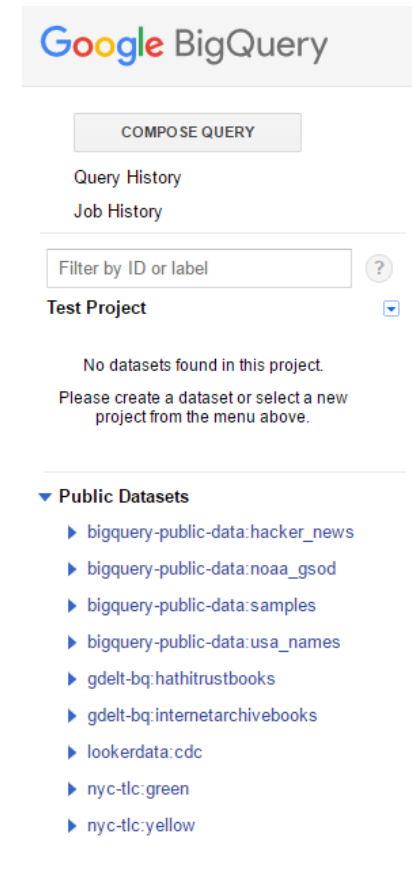
Home

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.



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.

The screenshot displays the Google BigQuery web interface. The top navigation bar includes the Google BigQuery logo and a 'COMPOSE QUERY' button. Below this, there are links for 'Query History' and 'Job History', a search bar labeled 'Filter by ID or label', and a 'Test Project' dropdown menu. The main content area is divided into two sections. On the left, under 'Public Datasets', a list of datasets is shown, including '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', and 'nyc-tlc.yellow'. On the right, the 'New Query' section contains a SQL query:

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

 Below the query editor are buttons for 'RUN QUERY', 'Save Query', 'Save View', 'Format Query', and 'Show Options'. The 'RUN QUERY' button is highlighted in red. To the right of these buttons, it says 'Query complete (1.5s elapsed, 8.75 MB processed)'. Below the buttons, there are three tabs: 'Results', 'Explanation', and 'Job Information'. The 'Results' tab is selected, showing a table with 17 rows. The table has two columns: 'spc_common' and 'CNT'. The data is sorted in descending order of 'CNT'. The first three rows are: 'London planetree' with a count of 87014, 'honeylocust' with a count of 64264, and 'Callery pear' with a count of 58931. The bottom of the interface shows a 'Table' tab and a 'JSON' tab, with 'Table' selected. At the bottom right, it says 'First < Prev Rows 1 - 17 of 17'.

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

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.

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 ?

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

RUN QUERY Save Query Save View Format Query Show Options Query complete (0.8s elapsed, cached)

Results	Explanation	Job Information
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

Table JSON

First < Prev Rows 1

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.

New Query ?

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

RUN QUERY

Save Query

Save View

Format Query

Show Options

Query complete (2.5s elapsed, 23.2 MB processed)

Results

Explanation

Job Information

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

Table

JSON

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.

New Query ?

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

RUN QUERY

Save Query

Save View

Format Query

Show Options

Query complete (2.6s elapsed, 22.7 MB processed)

Results Explanation Job Information

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

Table JSON

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.

New Query ?

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

RUN QUERY ▼

Save Query

Save View

Format Query

Show Options

Query complete (4.7s elapsed)

Results

Explanation

Job Information

Row	zipcode	SMMA
-----	---------	------

Query returned zero records.

Table JSON

Thank You!