Name: Mattis Meeuwesse

Student number: 622948

URL Github: https://github.com/Maestro1334/Hadoop

Assignment:

To get a 6

Count the number of ratings given for each movie

Code have to run on HDFS with a basic setup (Lecture 1 & 2)

Code is executed with a python command with MrJob

Upload a document on Moodle with the following information:

Name and student number

URL from GitHub with your source code Explain in the document what steps have to be taken to execute the code

Explain in your own words every step included source code.

Make a screenshot from the result and include it in your document

To get an 8:

Same as to get a 6, but additionally:

Sort the movies by their numbers of ratings

To get a 10:

Same as to get an 8, but additionally:

Sort the genres by their total ratings

Preparation

Steps before executing script:

- Make sure set up is **complete and functional** according to slides of lesson 1 + 2:
    - Hortonworks Sandbox HDP 2.6.5 is installed and running.
    - MrJob, Nano, Python are installed and working.
    - There is an active connection to the VM through a session by either PuTTy or MobaXterm.
    - Any other requirement described in slides lessons 1 + 2.
- Make sure the correct u.data file is present in directory of the script
- To execute the script, type in the CLI: “python sum_sort_rating.py u.data” and press enter.

Additionally, I personally have to execute the commands below in the CLI as a su root, EVERY SESSION. If the session for whatever reason disconnects or is restarted, these steps have to be executed again or MrJob will not work (for me). If you run into the problem of MrJob not working, you may have to execute these commands in the CLI as a super user, connected to the VM, first.

- Yum install scl-utils
- Yum install centos-release-scl
- Yum install python27
- Scl enable python27 bash

The output below is the result of running the script: A list of movie id's and their counted ratings, sorted by number of ratings in descending order. Number of ratings left, movie id right.

```
[root@sandbox-hdp maria_dev]# python sum_sort_rating.py u.data
No configs found; falling back on auto-configuration
Creating temp directory /tmp/sum_sort_rating.maria_dev.20210511.152554.708614
Running step 1 of 2...
Running step 2 of 2...
Streaming final output from /tmp/sum_sort_rating.maria_dev.20210511.152554.708614/
output...
2541    "50"
2111    "100"
2032    "181"
1936    "258"
1786    "174"
1769    "127"
1759    "286"
1753    "1"
1673    "98"
1645    "288"
1600    "56"
1565    "300"
1543    "172"
1531    "294"
1489    "7"
1486    "313"
1475    "121"
1425    "237"
1396    "117"
1359    "79"
1352    "173"
1342    "204"
1336    "222"
1331    "318"
1301    "210"
1285    "168"
1258    "64"
1237    "69"
1237    "269"
1236    "302"
1233    "22"
1184    "195"
1184    "151"
1182    "96"
1174    "183"
1171    "12"
1165    "9"
1150    "423"
1149    "191"
1140    "405"
1138    "89"
1135    "257"
1134    "216"
1133    "357"
1121    "176"
1107    "15"
1102    "276"
1085    "28"
1083    "483"
1075    "275"
1057    "234"
1054    "202"
```

Creating the script

The first step of the script is to import the required MrJob libraries.

```
1    from mrjob.job import MRJob
2    from mrjob.step import MRStep
```

Second step is to define a class. This class inherits from MrJob and will be used to define the steps of the job.

```
5    class RatingsSum(MRJob):
```

Third step is to define our own mapper and reducers.

The mapper takes three arguments, besides self (standard in python) it needs a key and a record (line). Then we yield the movie ID and ratings as data to work with.
In this case we ignore the key, and a single line of text input is the value.

```
14       def mapper_get_ratings(self, _, line):
15           (userID, movieID, rating, timestamp) = line.split('\t')
16           yield movieID, int(rating)
```

Sum and sort ratings

Then we define a reducer to sum the ratings. This reducer again takes three arguments: self, key (movieID in this case) and values (ratings). It then yields a pair of the sum of the values for each key (i.e. movieID) and the respective movieID.

In the first reducer, a pair is created to be processed by the next reducer, which sorts the pairs. The second reducer goes over each pair and sorts them to be yielded in order of number of ratings per movie in descending order.

```
18      def reducer_sum_ratings(self, key, values):
19          yield None, (sum(values), key)
20
21      def reducer_sort_ratings(self, _, pair):
22          sorted_pairs = sorted(pair, reverse=True)
23          for pair in sorted_pairs:
24              yield pair
```

Make executable from CLI

Lastly this piece of code is used to be able to execute the script in the command line. It executes the code in the specified and previously created class.

```
26
27  if __name__ == '__main__':
28      RatingsSum.run()
29
```