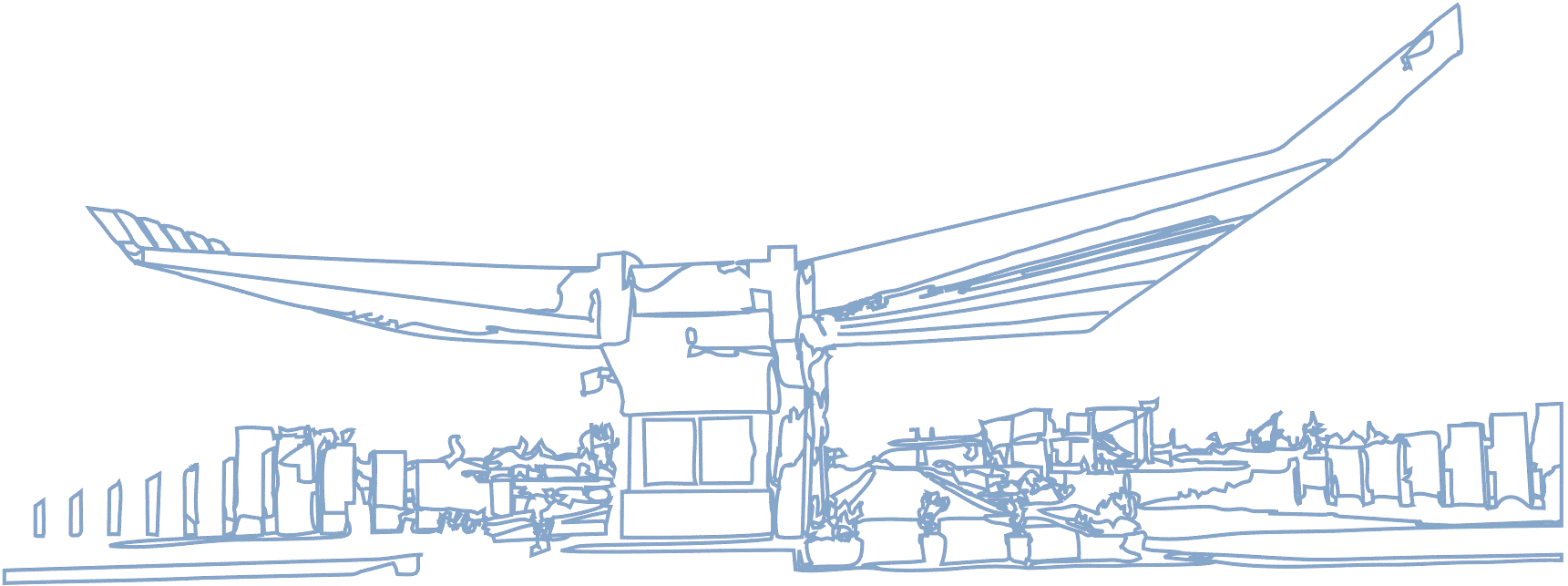
****

****

**Data Analysis with MapReduce and (Pig or Hive)**

**Project Documentation**

**31.05.2020**

Epoka University

Tirana, ALBANIA

PREPARED:

**Baftjar TABAKU**

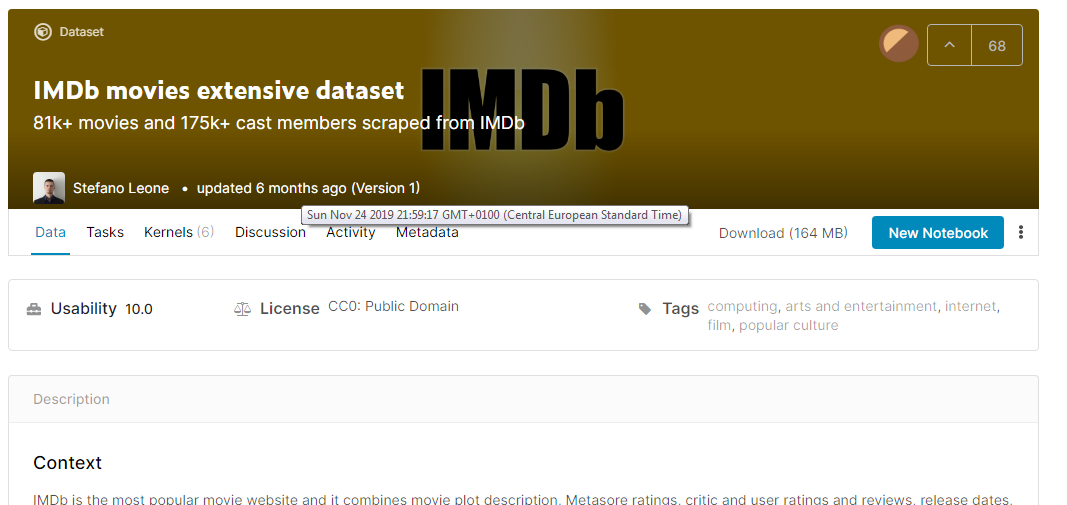
ACCEPTED:

**Prof.Dr. Arben Asllani**

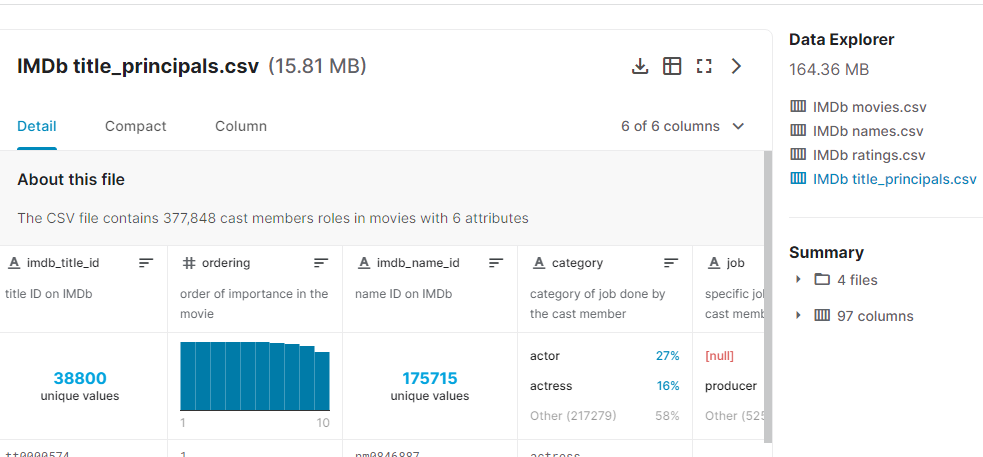
**CEN 571 – Data Mining**

1. **Dataset**

The dataset was taken by Kaggle.com



With a size of 168 MB, the latest one, of 6 months, composed of 4 tables, by Stefano Leone. All in CSV format as shown below.



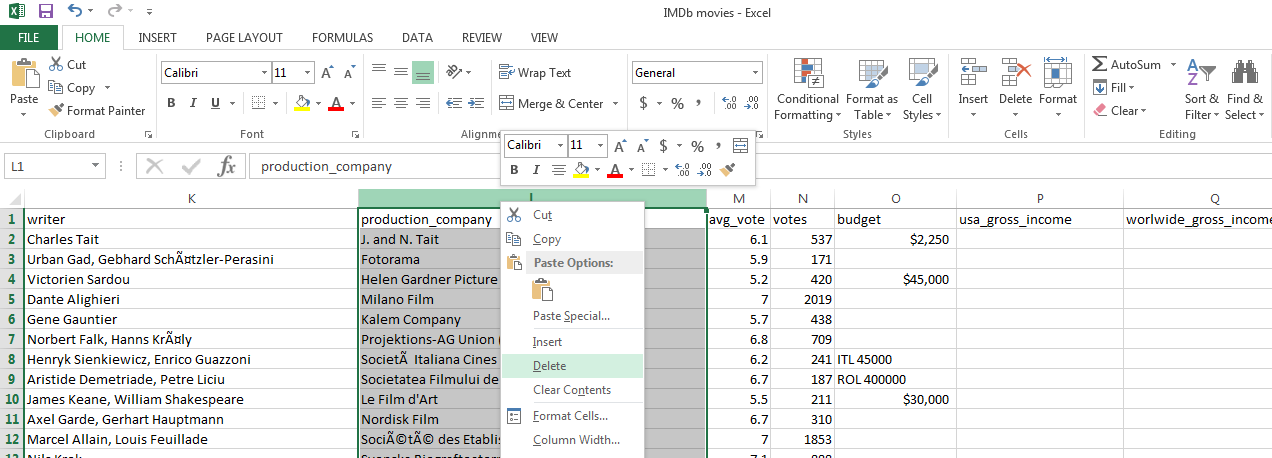
1. **Cleaning the unwanted data, the data selection according to the projects goals.**

Some redundant features will be removed, and data will be processed using Map Reduce, with corresponding code and jar files.

Removing some columns was used the Microsoft Excel, where the data was displayed better and modified.

Ex: According to my goals, I don’t need a movie Description, cast list, reviews numbers from users and anything to do with the price, or the user’s that rate professions, spouses number and so on.

We also delete the cast data from dataset.



From all data of 4 tables, I reduced it to 2 and removed the unnecessary features for all of them.

**Note: I could have done them in MapReduce too by just ignoring their position, but data was very large and hard to manage according my goals.**

1. **Writing the map reduce code**

By analyzing the data, for each movie, it was on my interest goal to see the difference of rating between males and females, I modified a map reduce code according to my dataset and what I want. A jar file was also exported for later use.

**import** java**.**io**.**IOException**;**

**import** java**.**util**.\*;**

**import** org**.**apache**.**hadoop**.**fs**.**Path**;**

**import** org**.**apache**.**hadoop**.**io**.\*;**

**import** org**.**apache**.**hadoop**.**mapreduce**.\*;**

**import** org**.**apache**.**hadoop**.**conf**.\*;**

**import** org**.**apache**.**hadoop**.**mapreduce**.**lib**.**input**.**FileInputFormat**;**

**import** org**.**apache**.**hadoop**.**mapreduce**.**lib**.**output**.**FileOutputFormat**;**

public class TotalOpinionDifference **{**

public static class MyMapper **extends** Mapper**<**LongWritable**,** Text**,** Text**,** DoubleWritable**>** **{**

private Text imdb\_title\_id **=** **new** Text**();**

private DoubleWritable opinion\_difference **=** **new** DoubleWritable**(**1**);**

public void map**(**LongWritable key**,** Text value**,** Context context**)** **throws** IOException**,** InterruptedException **{**

String line **=** value**.**toString**();**

String**[]** tokens **=** line**.**split**(**","**);**

**{**

**try** **{**

float males\_allages\_avg\_vote **=** Float**.**parseFloat**(**tokens**[**5**]);**

float females\_allages\_avg\_vote **=** Float**.**parseFloat**(**tokens**[**6**]);**

float avg\_diference **=** Math**.**abs**(**males\_allages\_avg\_vote**-**females\_allages\_avg\_vote**);** //difference rating

imdb\_title\_id**.**set**(**tokens**[**0**]);**

opinion\_difference**.**set**(**avg\_diference**);**

context**.**write**(**imdb\_title\_id**,** opinion\_difference**);**

**}** **catch** **(**Exception ex**)** **{**

System**.**out**.**println**(**imdb\_title\_id**);**

System**.**out**.**println**(**ex**.**toString**());**

**}**

**}**

**}**

**}**

public static class MyReducer **extends** Reducer**<**Text**,** DoubleWritable**,** Text**,** DoubleWritable**>** **{**

public void reduce**(**Text key**,** Iterable**<**DoubleWritable**>** values**,** Context context**)**

**throws** IOException**,** InterruptedException **{**

double sum **=** 0**;**

int count **=** 0**;**

**for** **(**DoubleWritable val **:** values**)** **{**

sum **=** sum **+** val**.**get**();**

count**++;**

**}**

double rate\_dif\_avg **=** 0**;**

rate\_dif\_avg **=** sum **/** count**;**

context**.**write**(**key**,** **new** DoubleWritable**(**rate\_dif\_avg**));**

**}**

**}**

public static void main**(**String**[]** args**)** **throws** Exception **{**

Configuration conf **=** **new** Configuration**();**

Job job **=** **new** Job**(**conf**,** "TotalOpinionDifference"**);**

job**.**setJarByClass**(**TotalOpinionDifference**.**class**);**

job**.**setMapperClass**(**MyMapper**.**class**);**

job**.**setReducerClass**(**MyReducer**.**class**);**

job**.**setMapOutputKeyClass**(**Text**.**class**);**

job**.**setMapOutputValueClass**(**DoubleWritable**.**class**);**

job**.**setOutputKeyClass**(**Text**.**class**);**

job**.**setOutputValueClass**(**IntWritable**.**class**);**

FileInputFormat**.**addInputPath**(**job**,** **new** Path**(**args**[**0**]));**

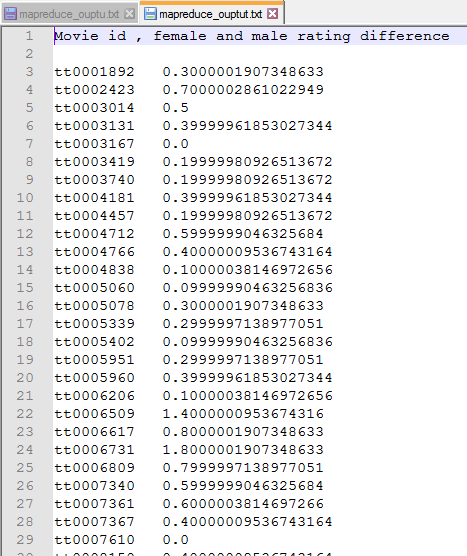
FileOutputFormat**.**setOutputPath**(**job**,** **new** Path**(**args**[**1**]));**

job**.**waitForCompletion**(true);**

**}**

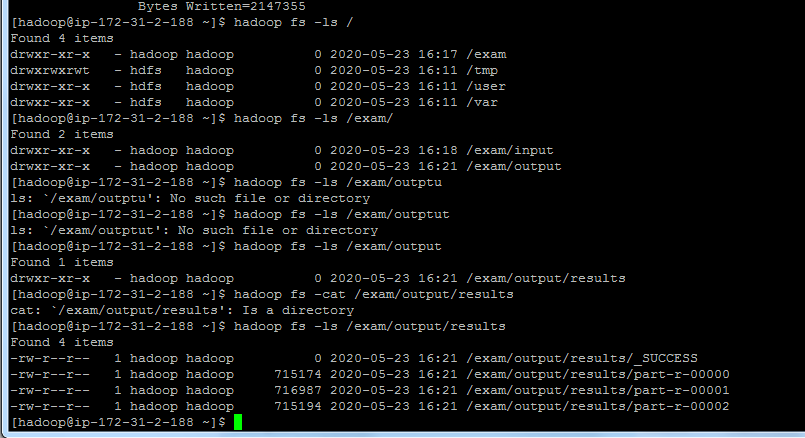
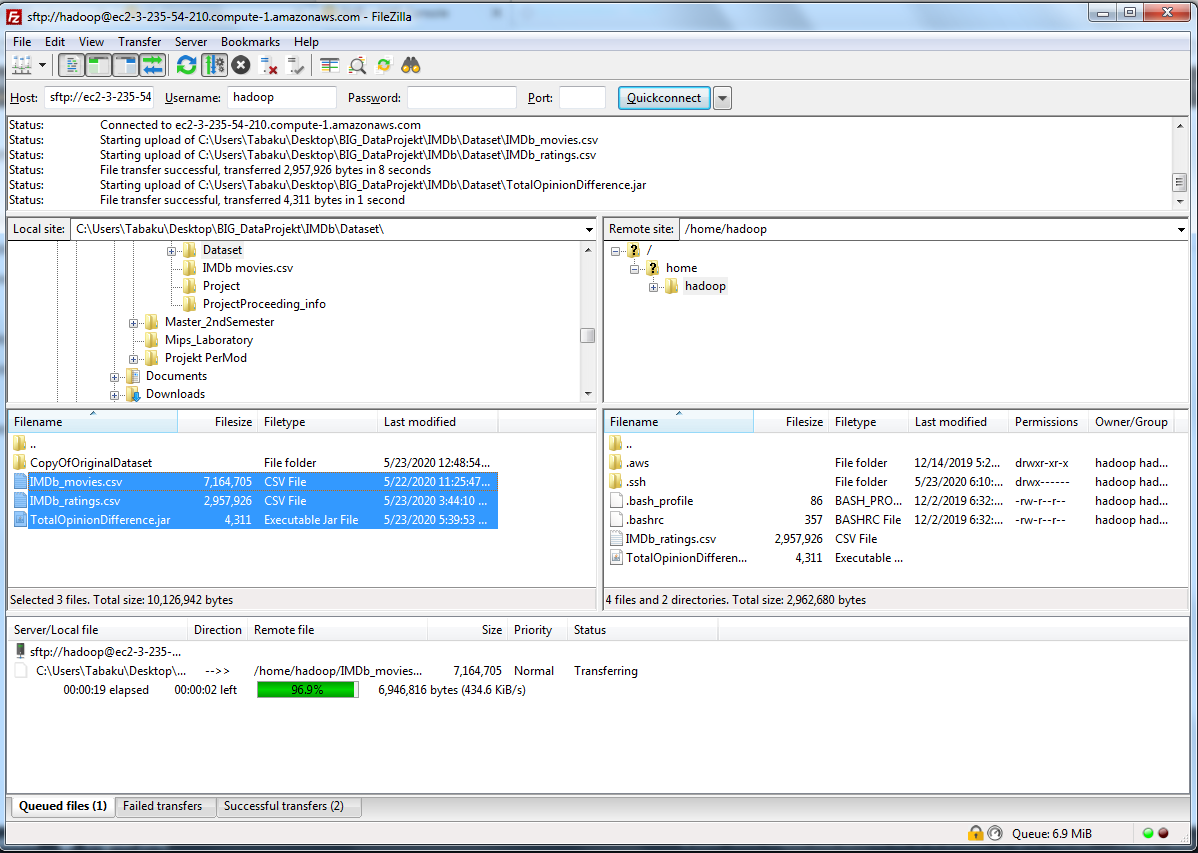
**}**

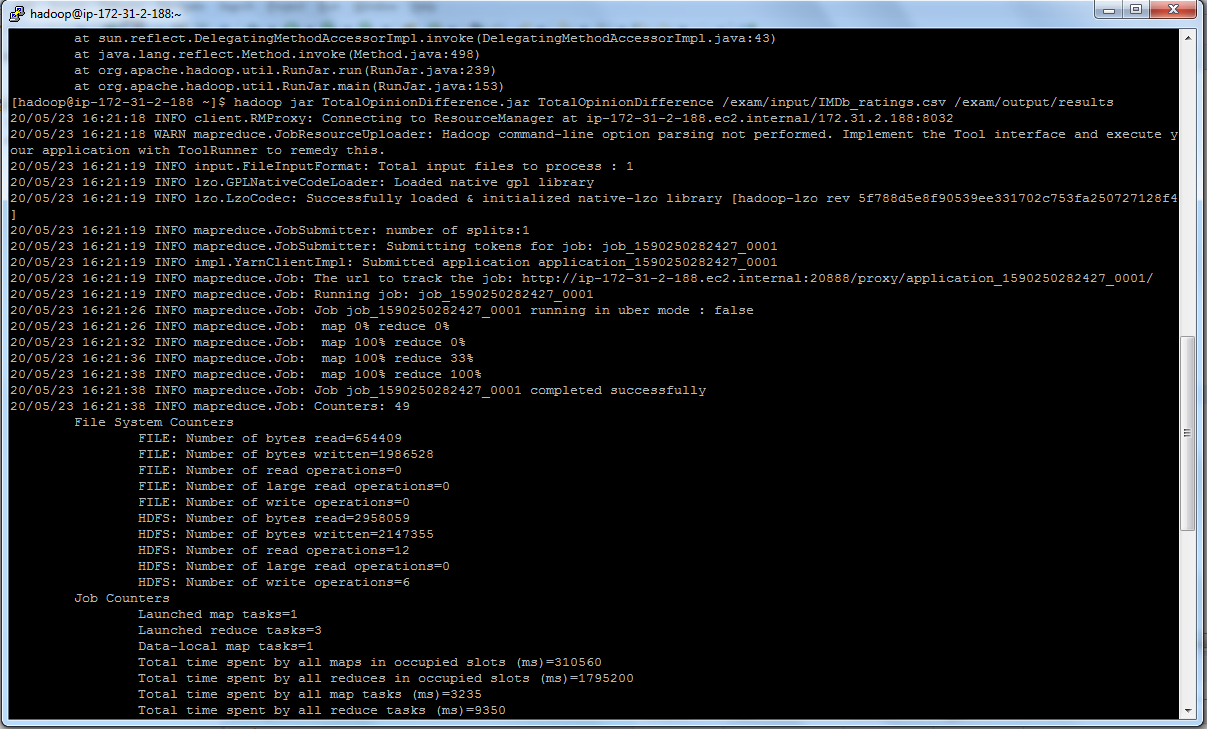
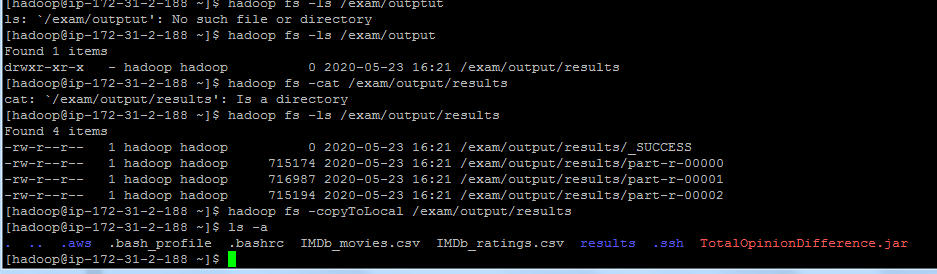
Output part,

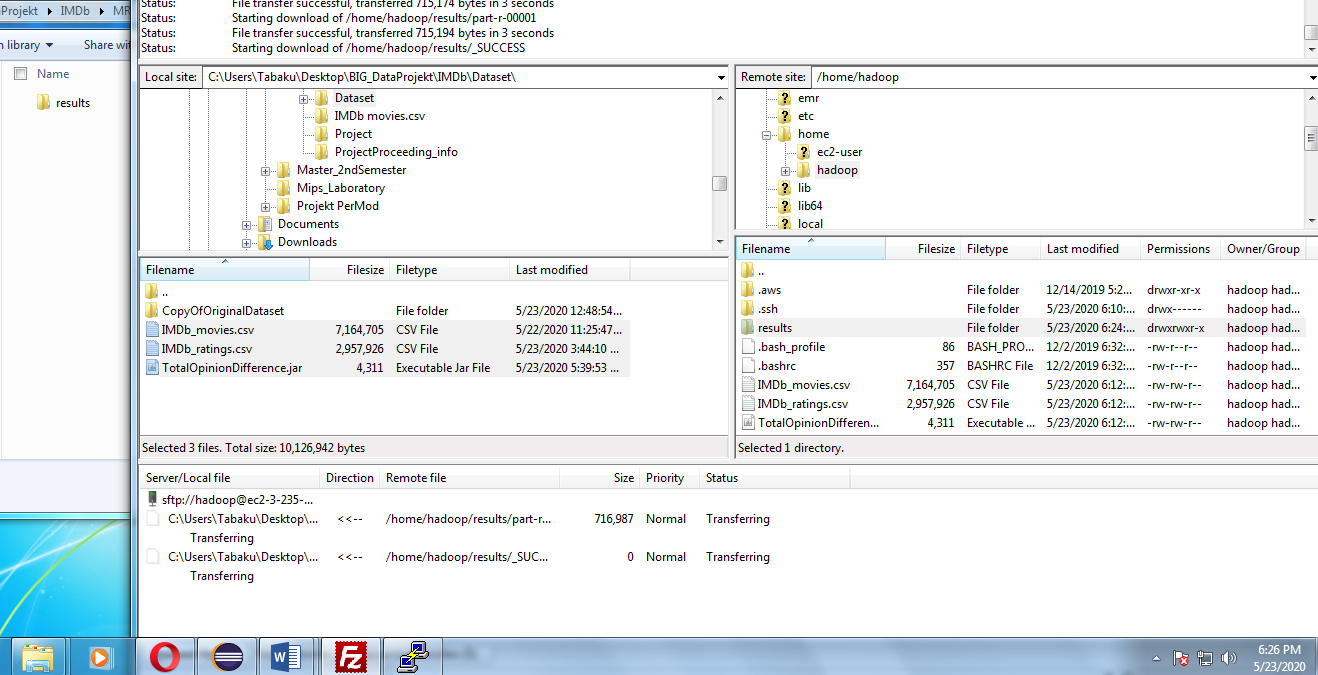


1. **Uploading the data into the cluster and processing calculations.**

The data tables.csv were successfully added to the AWS cluster for further analyzing, using MapReduce and Hive.

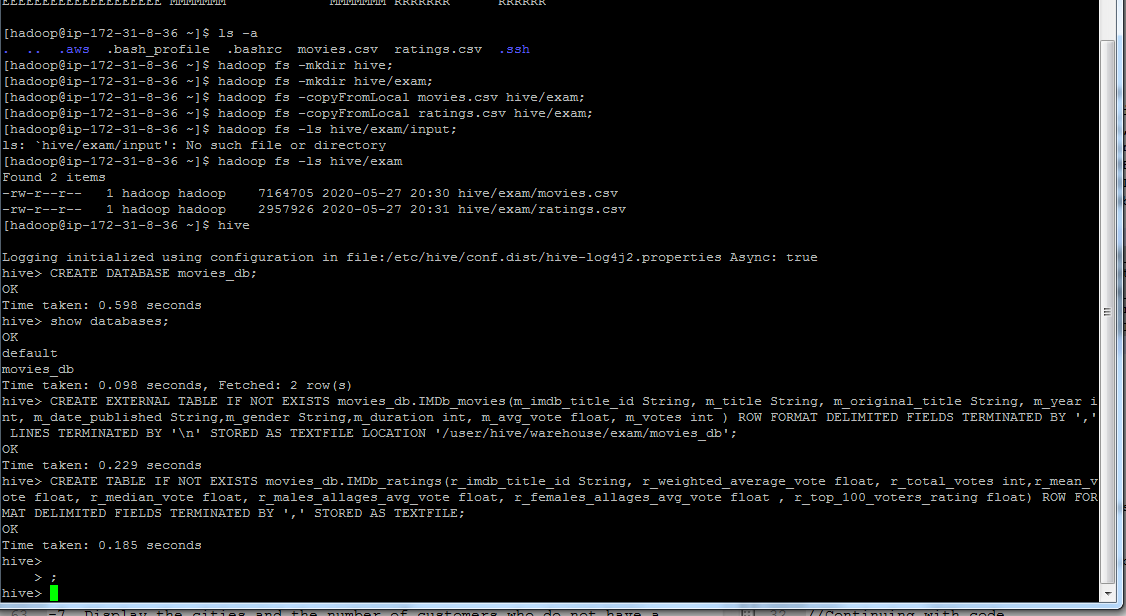
****

****

****

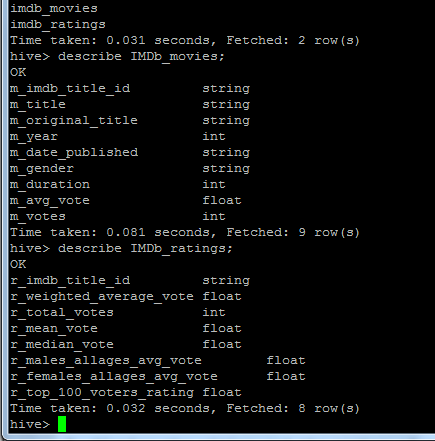
The MapReduce output will be stored in the output folder, according to the ordering of the submission, below are the corresponding output screenshots.

1. **Hive used commands and their output**

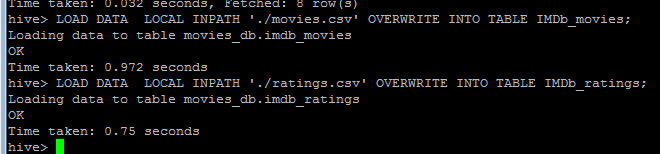
I choose HIVE because the characteristic of my project goals are the reports and data analysis, rather than developing and programming.

The data successfully moved into the Hadoop and the hive databases and tables are successfully created.

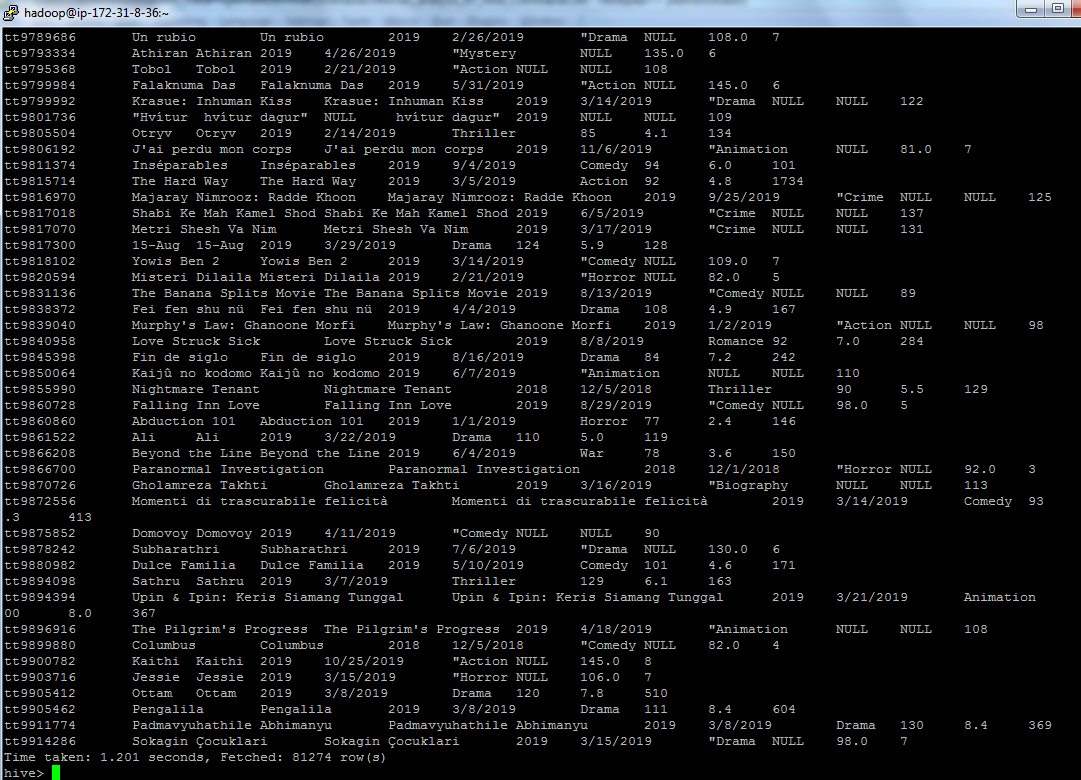
Describing the data structure of the table



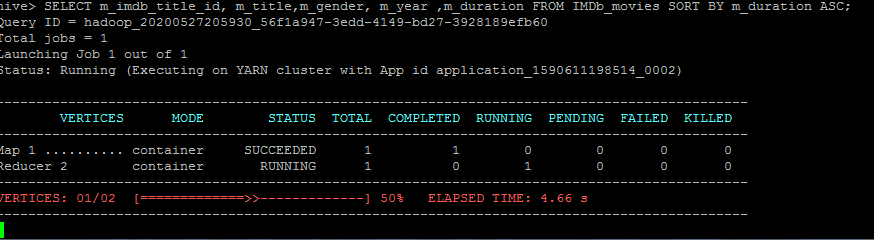
Populating the data into the respective tables

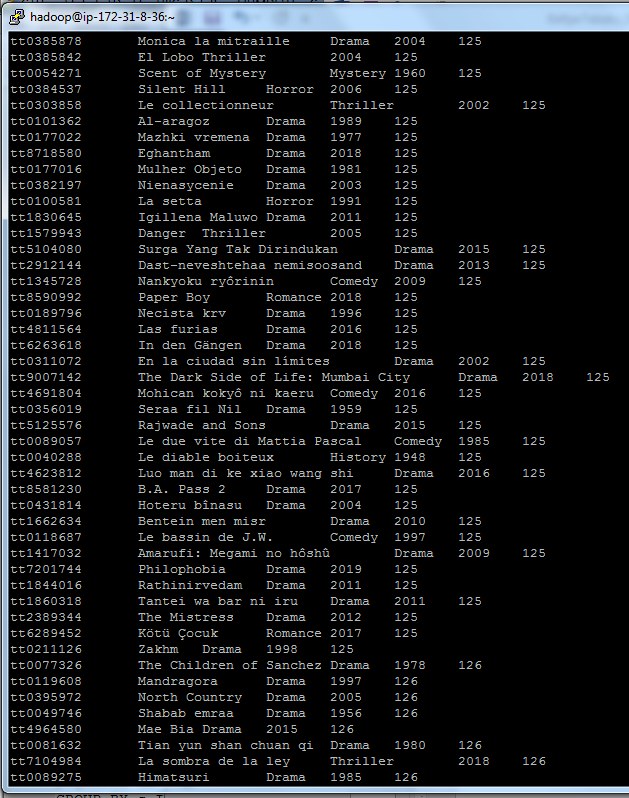


Selecting all the movie data.

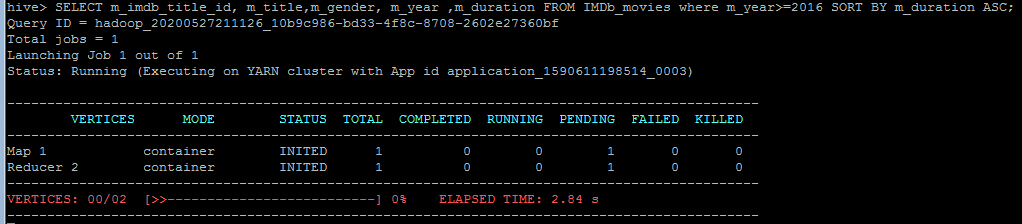


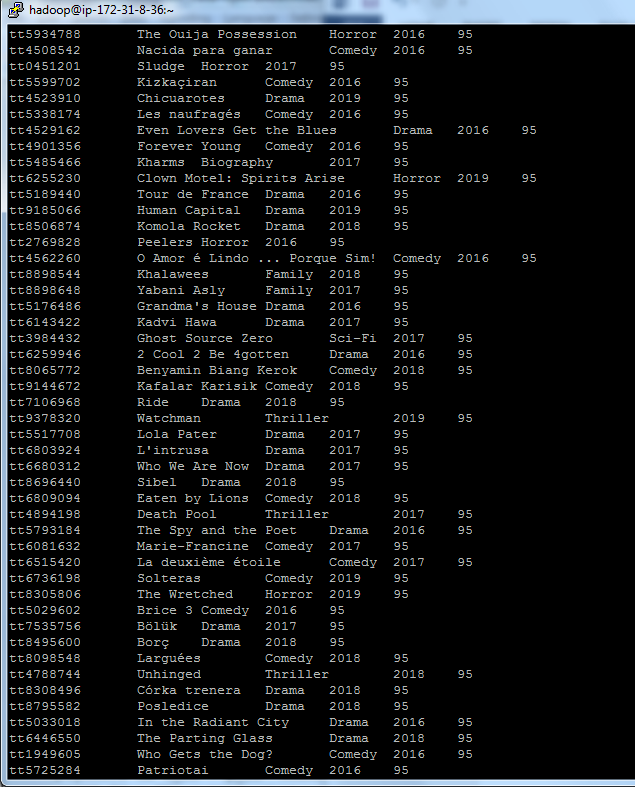
Sorting the movies according to the duration,



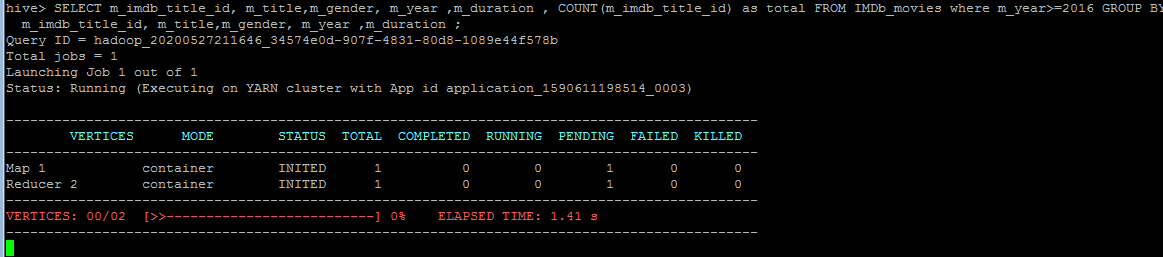


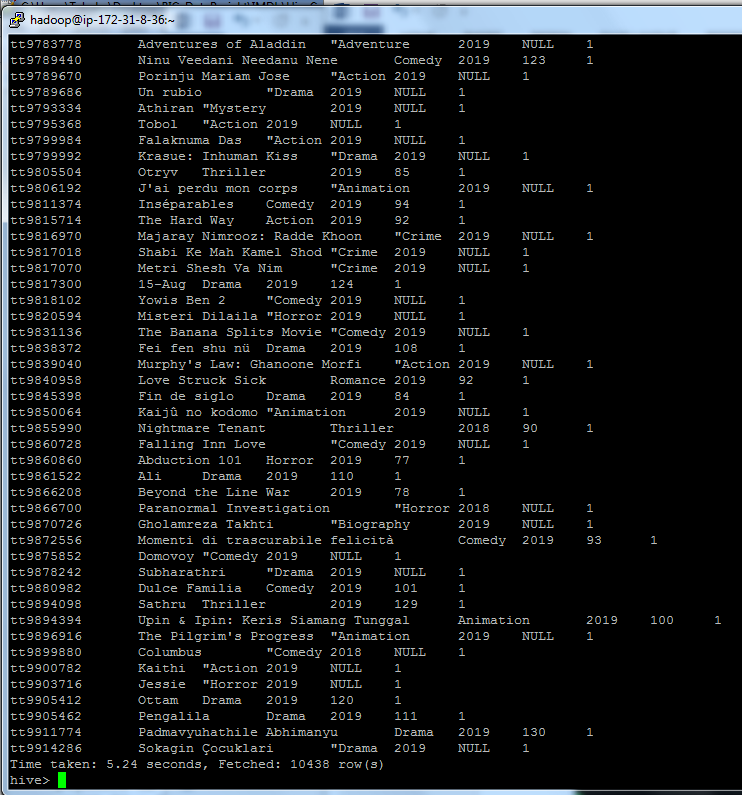
Selecting all the movies from year 2016, and analyzing according to their duration.





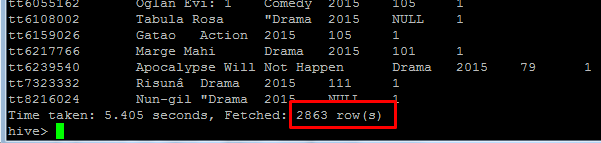
Counting the movies from 2016 until today,



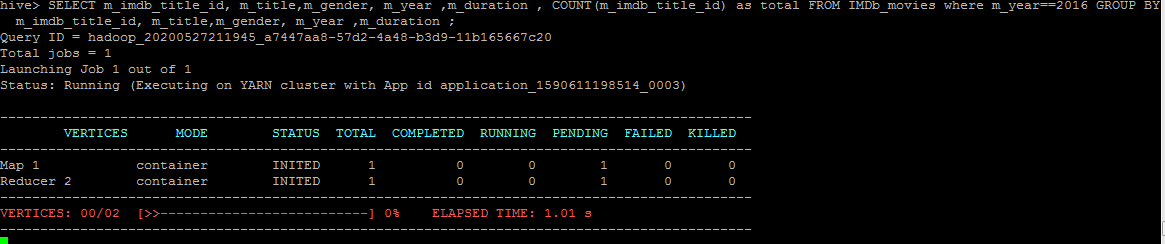


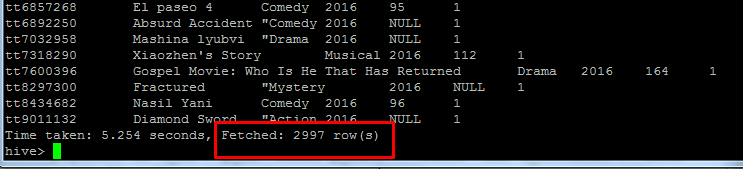
Counting the movies for year 2015, 2016, 2017, and 2018.

2015 movies counted,

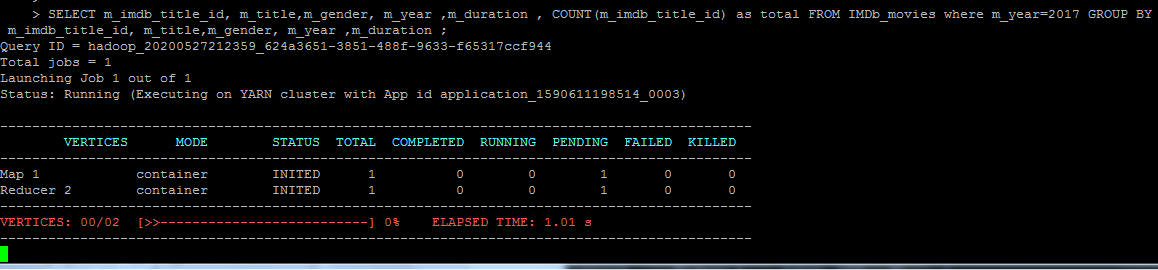


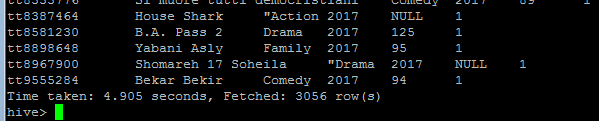
2016 movies counted,



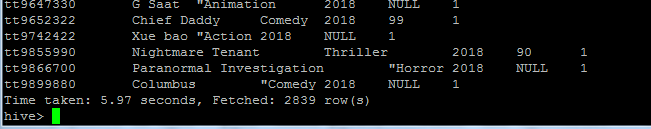


2017 movies counted,



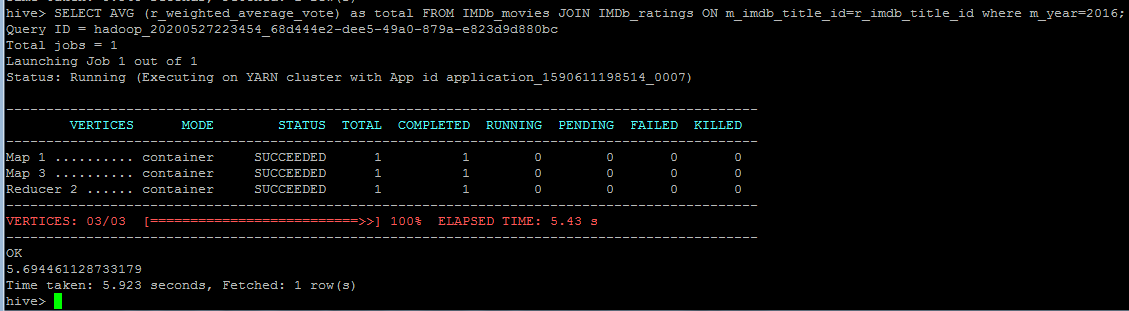


2018 movies counted,

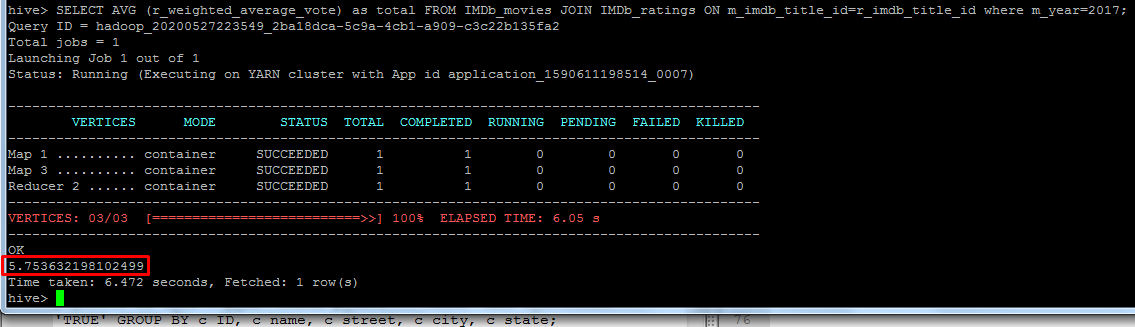


The movies total average ratting, according to the years (2016, 2017, and 2018).

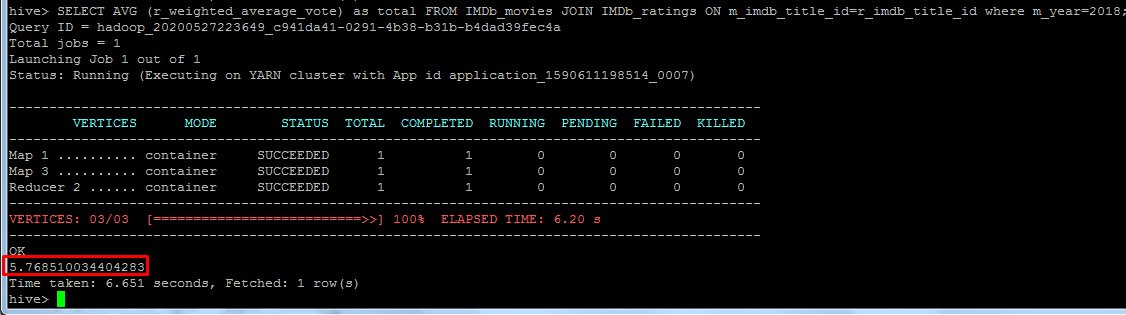
The year 2016.



The year 2017.

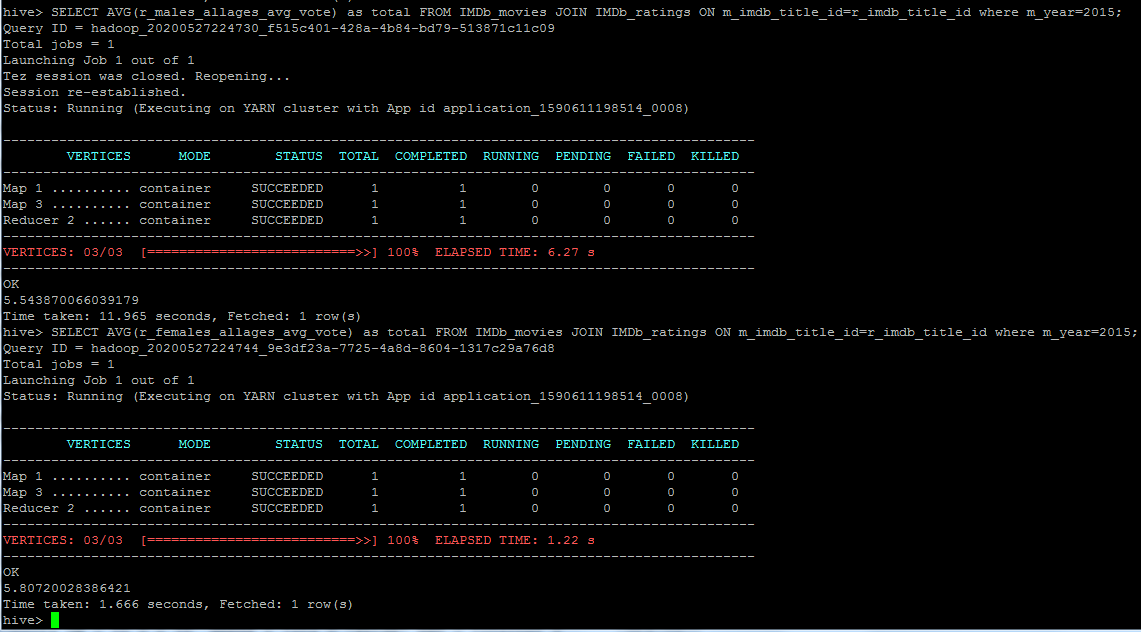


The year 2018.

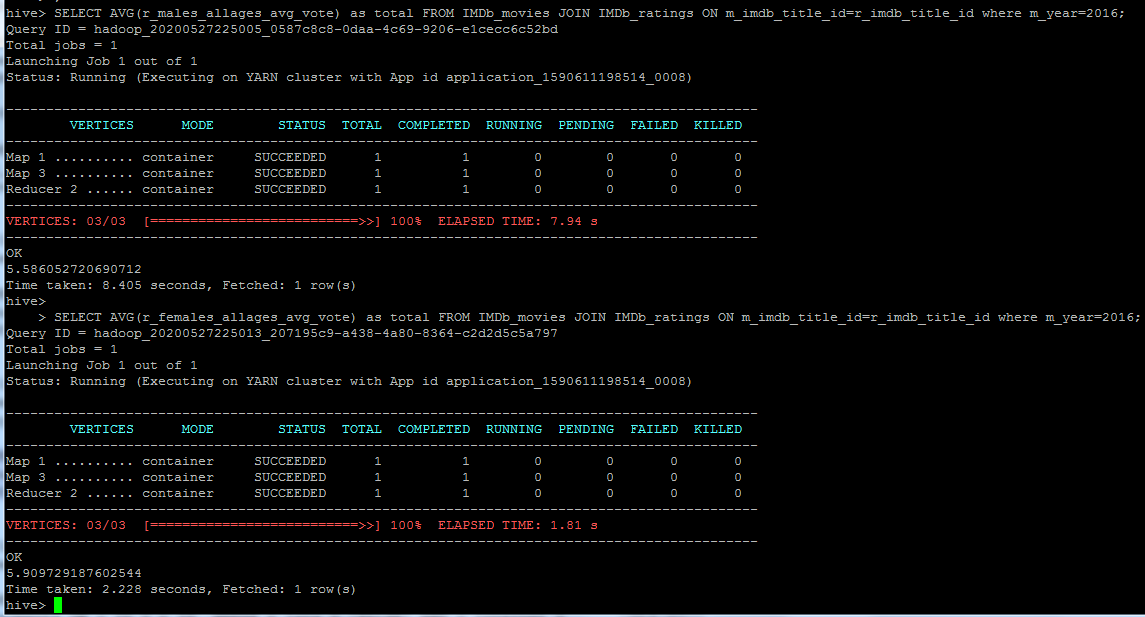
.

Males and females rating for the years 2015,2016,2017,2018,

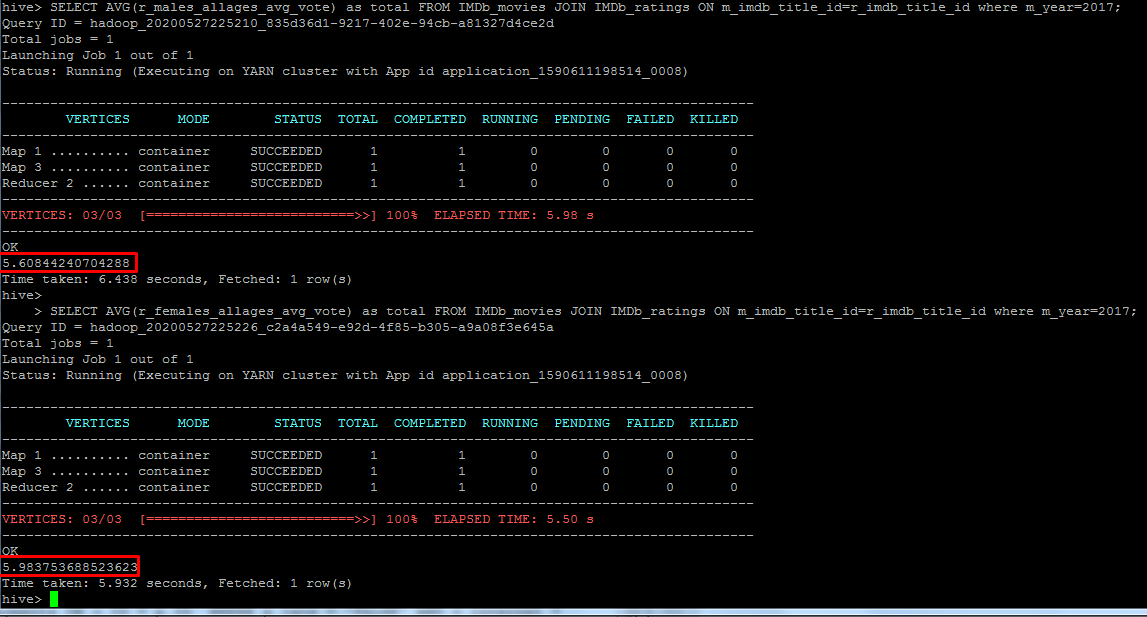
Year 2015, males and females rating average



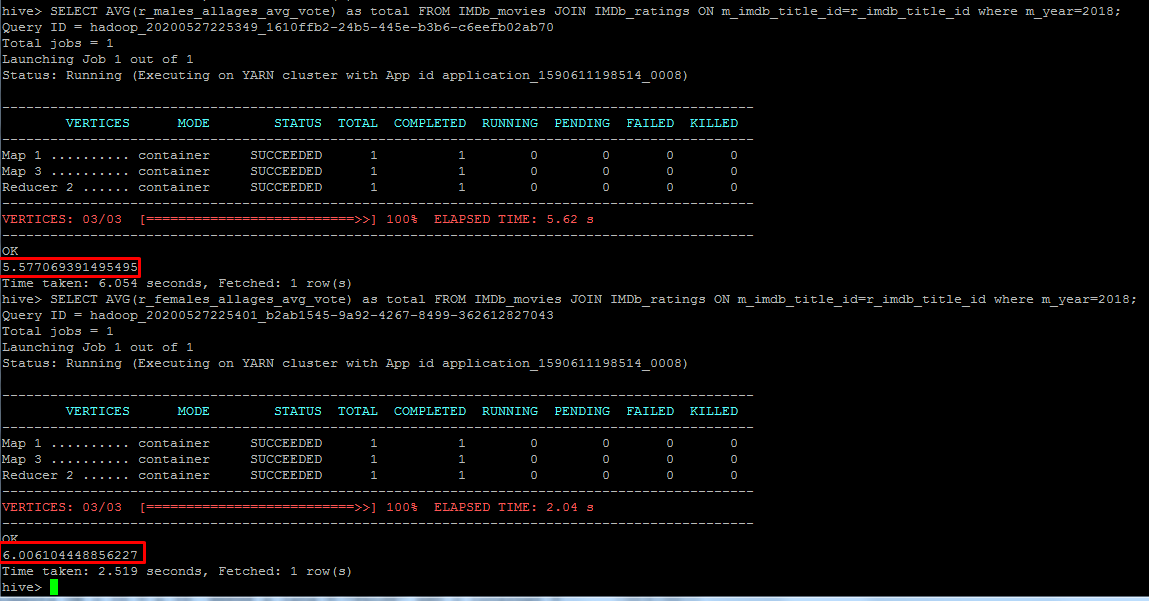
Year 2016, males and females,



Year 2017, males and females.



Year 2018, males and females votes average.



**Note: These queries do not show all kind of operations, ex. Count, and so on, the queries are performed according to the goals of the project only.**

**Queries**

hadoop fs -mkdir hive;

hadoop fs -mkdir hive/exam;

hadoop fs -mkdir hive/exam/input;

hadoop fs -copyFromLocal movies.csv hive/exam/input;

hadoop fs -copyFromLocal ratings.csv hive/exam/input;

hadoop fs -ls hive/exam/input;

hive

CREATE DATABASE movies\_db;

show databases;

CREATE EXTERNAL TABLE IF NOT EXISTS movies\_db.IMDb\_movies(m\_imdb\_title\_id String, m\_title String, m\_original\_title String, m\_year int, m\_date\_published String,m\_gender String,m\_duration int, m\_avg\_vote float, m\_votes int ) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n' STORED AS TEXTFILE LOCATION '/user/hive/warehouse/exam/movies\_db';

CREATE TABLE IF NOT EXISTS movies\_db.IMDb\_ratings(r\_imdb\_title\_id String, r\_weighted\_average\_vote float, r\_total\_votes int,r\_mean\_vote float, r\_median\_vote float, r\_males\_allages\_avg\_vote float, r\_females\_allages\_avg\_vote float , r\_top\_100\_voters\_rating float) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE;

use movies\_db;

show tables;

describe IMDb\_movies;

describe IMDb\_ratings;

//load data

LOAD DATA LOCAL INPATH './movies.csv' OVERWRITE INTO TABLE IMDb\_movies;

LOAD DATA LOCAL INPATH './ratings.csv' OVERWRITE INTO TABLE IMDb\_ratings;

//Continuing with code

-Sorting the movies according to lengths

SELECT \* FROM IMDb\_movies;

SELECT \* FROM IMDb\_ratings;

//Sorting all the movies according to the time

SELECT m\_imdb\_title\_id, m\_title,m\_gender, m\_year ,m\_duration FROM IMDb\_movies SORT BY m\_duration ASC;

//selecting movies from year 2016 to current

SELECT m\_imdb\_title\_id, m\_title,m\_gender, m\_year ,m\_duration FROM IMDb\_movies where m\_year>=2016 SORT BY m\_duration ASC;

//counting movies from year 2016 and above

SELECT m\_imdb\_title\_id, m\_title,m\_gender, m\_year ,m\_duration , COUNT(m\_imdb\_title\_id) as total FROM IMDb\_movies where m\_year>=2016 GROUP BY m\_imdb\_title\_id, m\_title,m\_gender, m\_year ,m\_duration ;

//counting movies of year 2016

SELECT m\_imdb\_title\_id, m\_title,m\_gender, m\_year ,m\_duration , COUNT(m\_imdb\_title\_id) as total FROM IMDb\_movies where m\_year=2016 GROUP BY m\_imdb\_title\_id, m\_title,m\_gender, m\_year ,m\_duration ;

//counting movies of year 2017

SELECT m\_imdb\_title\_id, m\_title,m\_gender, m\_year ,m\_duration , COUNT(m\_imdb\_title\_id) as total FROM IMDb\_movies where m\_year=2017 GROUP BY m\_imdb\_title\_id, m\_title,m\_gender, m\_year ,m\_duration ;

//counting movies of year 2018

SELECT m\_imdb\_title\_id, m\_title,m\_gender, m\_year ,m\_duration , COUNT(m\_imdb\_title\_id) as total FROM IMDb\_movies where m\_year=2018 GROUP BY m\_imdb\_title\_id, m\_title,m\_gender, m\_year ,m\_duration ;

//Calculating the rating average of the movies for the year 2016, 2017,2018

//year 2016 average ratting

SELECT AVG (r\_weighted\_average\_vote) as total FROM IMDb\_movies JOIN IMDb\_ratings ON m\_imdb\_title\_id=r\_imdb\_title\_id where m\_year=2016;

//year 2017 average ratting

SELECT AVG (r\_weighted\_average\_vote) as total FROM IMDb\_movies JOIN IMDb\_ratings ON m\_imdb\_title\_id=r\_imdb\_title\_id where m\_year=2017;

//year 2018 average ratting

SELECT AVG(r\_weighted\_average\_vote) as total FROM IMDb\_movies JOIN IMDb\_ratings ON m\_imdb\_title\_id=r\_imdb\_title\_id where m\_year=2018;

//Total Males and females ratting averafe for years 2015,2016,2017,2018

//females

SELECT AVG(r\_females\_allages\_avg\_vote) as total FROM IMDb\_movies JOIN IMDb\_ratings ON m\_imdb\_title\_id=r\_imdb\_title\_id where m\_year=2015;

//mles

SELECT AVG(r\_males\_allages\_avg\_vote) as total FROM IMDb\_movies JOIN IMDb\_ratings ON m\_imdb\_title\_id=r\_imdb\_title\_id where m\_year=2015;

//2016

//mles

SELECT AVG(r\_males\_allages\_avg\_vote) as total FROM IMDb\_movies JOIN IMDb\_ratings ON m\_imdb\_title\_id=r\_imdb\_title\_id where m\_year=2016;

//females

SELECT AVG(r\_females\_allages\_avg\_vote) as total FROM IMDb\_movies JOIN IMDb\_ratings ON m\_imdb\_title\_id=r\_imdb\_title\_id where m\_year=2016;

//2017

//mles

SELECT AVG(r\_males\_allages\_avg\_vote) as total FROM IMDb\_movies JOIN IMDb\_ratings ON m\_imdb\_title\_id=r\_imdb\_title\_id where m\_year=2017;

//females

SELECT AVG(r\_females\_allages\_avg\_vote) as total FROM IMDb\_movies JOIN IMDb\_ratings ON m\_imdb\_title\_id=r\_imdb\_title\_id where m\_year=2017;

//2018

//mles

SELECT AVG(r\_males\_allages\_avg\_vote) as total FROM IMDb\_movies JOIN IMDb\_ratings ON m\_imdb\_title\_id=r\_imdb\_title\_id where m\_year=2018;

//females

SELECT AVG(r\_females\_allages\_avg\_vote) as total FROM IMDb\_movies JOIN IMDb\_ratings ON m\_imdb\_title\_id=r\_imdb\_title\_id where m\_year=2018;