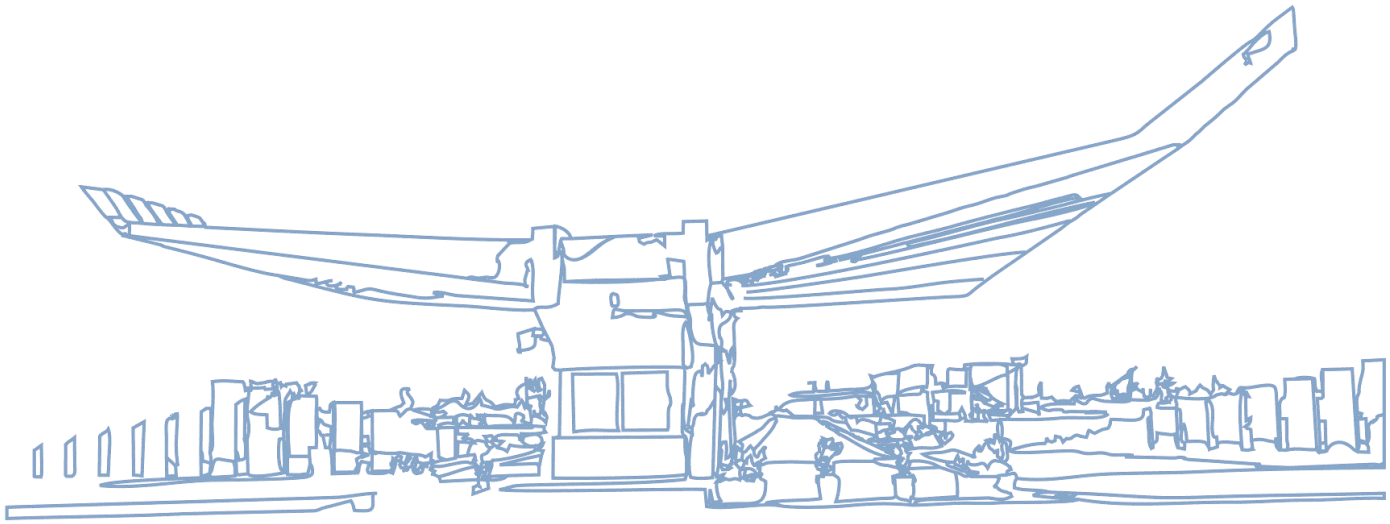
EPOKA UNIVERSITY | Computer Engineering Department

CEN 571 – Data Mining

Data Mining with Spark



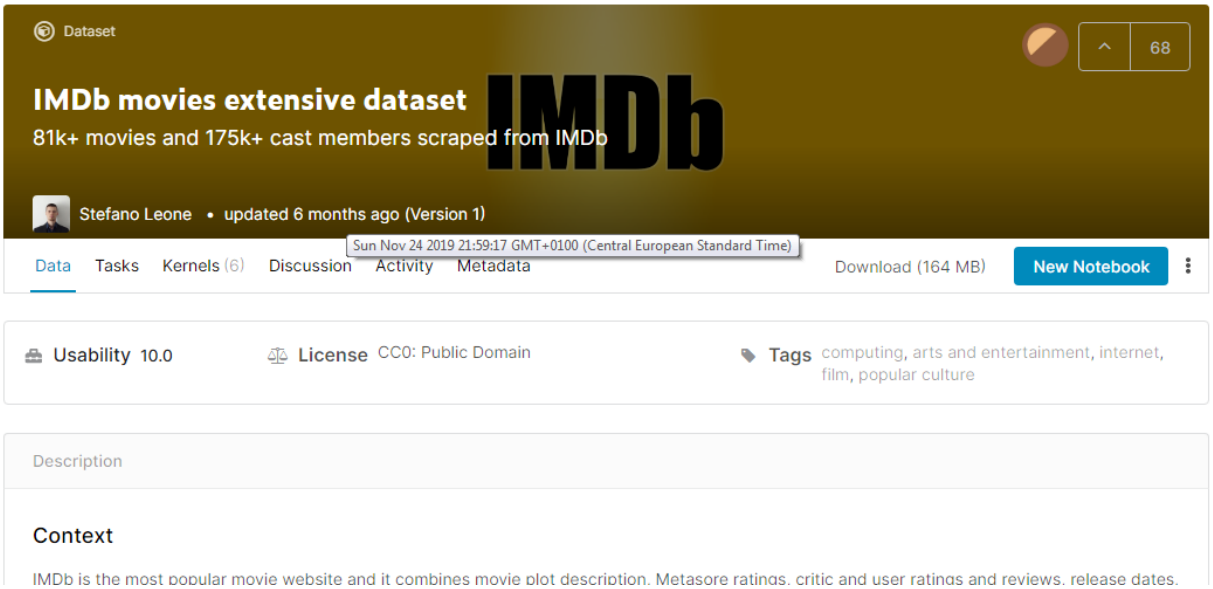
PREPARED:
Baftjar TABAKU

21.6.2020
Epoka University
Tirana, ALBANIA

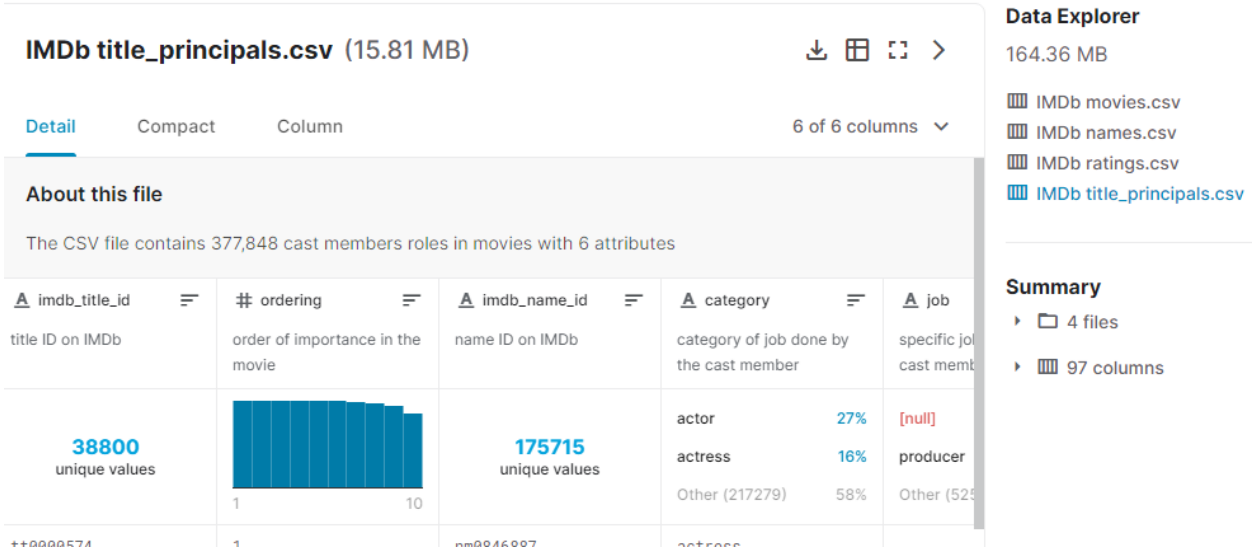
ACCEPTED:
Prof.Dr. Arben Asllani

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.



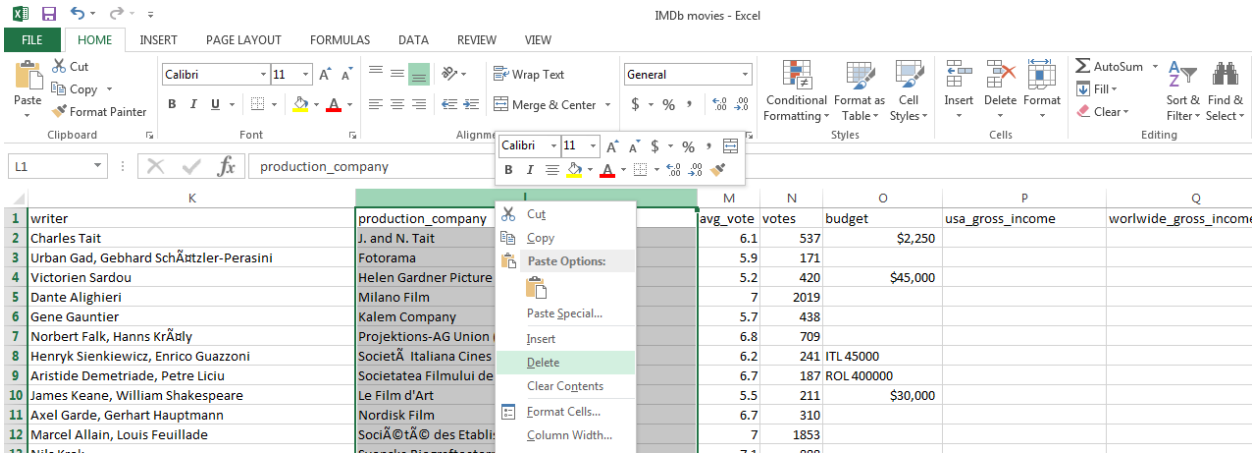
2. Cleaning the unwanted data, the data selection according to the project's goals.

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.



writer	production_company	avg_vote	votes	budget	usa_gross_income	worlwide_gross_income
Charles Tait	J. and N. Tait	6.1	537	$2,250		
Urban Gad, Gebhard SchÃ¤tzler-Perasini	Fotorama	5.9	171			
Victorien Sardou	Helen Gardner Picture	5.2	420	$45,000		
Dante Alighieri	Milano Film	7	2019			
Gene Gauntier	Kalem Company	5.7	438			
Norbert Falk, Hanns KrÃ¤ly	Projektions-AG Union	6.8	709			
Henryk Sienkiewicz, Enrico Guazzoni	SocietÃ Italiana Cines	6.2	241	ITL 45000		
Aristide Demetriade, Petre Liciu	Societatea Filmului de	6.7	187	ROL 400000		
James Keane, William Shakespeare	Le Film d'Art	5.5	211	$30,000		
Axel Garde, Gerhart Hauptmann	Nordisk Film	6.7	310			
Marcel Allain, Louis Feuillade	SociÃ©tÃ© des Etabli	7	1853			

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

3. Writing the Spark Application code in python

All the code is included as a single python file, included on the project submission folders.

```
"""
Main commands to proceed with, before the python

hadoop fs -ls
hadoop fs -mkdir /Spark_APP
hadoop fs -mkdir /Spark_APP/input
hadoop fs -copyFromLocal movies.csv /Spark_APP/input
hadoop fs -copyFromLocal ratings.csv /Spark_APP/input
hadoop fs -ls /Spark_APP/input

//run py Spark_APP
spark-submit IBDM_movies.py /Spark_APP/input/movies.csv
/Spark_APP/input/ratings.csv /Spark_APP
"""

"""
Author: Baftjar Tabaku
Epoka University
Data Mining
"""

import sys
from sched import scheduler

from pyspark.sql import SparkSession
from pyspark.sql.types import IntegerType
from pyspark.sql.types import FloatType

if __name__ == "__main__":
    if len(sys.argv) < 4:
        sys.stderr.write("Error: Usage: IBDM_movies.py <input-file 1> <input-
file 2> </Root of files>")
        sys.exit()

    spark = SparkSession.builder.getOrCreate()
    spark.sparkContext.setLogLevel("WARN")


    # Functions part to get the genders differences in vote for each movie
    def opinion_difference(num1, num2):
        return num1 - num2


    # Creating data frames
    # Movies table part, dealing with movies

    # IMDb_RDD_movies = spark.read.format("csv").option("header",
"true").load(
    #     "C:\\Users\\Baftjar
Tabaku\\PycharmProjects\\DataMining\\movies.csv")
    IMDb_RDD_movies = spark.read.format("csv").option("header",
"true").load(sys.argv[1])
```

```
    # cast each variable to the current data type , except the strings
    IMDb_RDD_movies = IMDb_RDD_movies.withColumn("year",
IMDb_RDD_movies["year"].cast(IntegerType()))
    IMDb_RDD_movies = IMDb_RDD_movies.withColumn("duration",
IMDb_RDD_movies["duration"].cast(IntegerType()))
    IMDb_RDD_movies = IMDb_RDD_movies.withColumn("avg_vote",
IMDb_RDD_movies["avg_vote"].cast(FloatType()))
    IMDb_RDD_movies = IMDb_RDD_movies.withColumn("votes",
IMDb_RDD_movies["votes"].cast(IntegerType()))

    # Second RDD Ratings part
    IMDb_RDD_ratings = spark.read.format("csv").option("header",
"true").load(sys.argv[2])
    # Casting the variables
    IMDb_RDD_ratings = IMDb_RDD_ratings.withColumn("weighted_average_vote",

IMDb_RDD_ratings["weighted_average_vote"].cast(FloatType()))
    IMDb_RDD_ratings = IMDb_RDD_ratings.withColumn("total_votes",

IMDb_RDD_ratings["total_votes"].cast(IntegerType()))
    IMDb_RDD_ratings = IMDb_RDD_ratings.withColumn("mean_vote",
IMDb_RDD_ratings["mean_vote"].cast(FloatType()))
    IMDb_RDD_ratings = IMDb_RDD_ratings.withColumn("median_vote",

IMDb_RDD_ratings["median_vote"].cast(IntegerType()))
    IMDb_RDD_ratings = IMDb_RDD_ratings.withColumn("males_allages_avg_vote",

IMDb_RDD_ratings["males_allages_avg_vote"].cast(FloatType()))
    IMDb_RDD_ratings =
IMDb_RDD_ratings.withColumn("females_allages_avg_vote",

IMDb_RDD_ratings["females_allages_avg_vote"].cast(FloatType()))
    IMDb_RDD_ratings = IMDb_RDD_ratings.withColumn("top1000_voters_rating",

IMDb_RDD_ratings["top1000_voters_rating"].cast(FloatType()))

    # custom operations with functions
    IMDb_RDD_ratings = IMDb_RDD_ratings.withColumn("opinion_diff",

opinion_difference(IMDb_RDD_ratings["males_allages_avg_vote"],

IMDb_RDD_ratings[

"females_allages_avg_vote"]).cast(
                                                    FloatType()))

    # IMDb_RDD_ratings = IMDb_RDD_ratings.withColumn("opinion_diff",
abs(["opinion_diff"]))

    # On the movies data we add a column, as the first task, the difference
in opinion

    IMDb_RDD_movies.printSchema()
    IMDb_RDD_movies.show()

    IMDb_RDD_ratings.printSchema()
    IMDb_RDD_ratings.show()

    # ----------------------------------------------------------------
    # Operations part, making them as tables for further operations
```

```
    # Creating two main tables
    IMDb_RDD_movies.registerTempTable("IMDb_movies")
    IMDb_RDD_ratings.registerTempTable("IMDb_ratings")

    selected_all_movies = spark.sql("SELECT * FROM IMDb_movies")
    selected_all_ratings = spark.sql("SELECT * FROM IMDb_ratings")
    selected_all_movies.show(10)
    selected_all_ratings.show(10)

    # Sort all movies
    selected_all_movies = spark.sql(
        "SELECT imdb_title_id, title,genre, year ,duration FROM IMDb_movies
SORT BY duration ASC")
    # selected_all_movies.show(100)  # first 100 movies sorted to demonstrate
the query
    selected_all_movies.show(10)

    # counting movies of year 2010
    count_2010_movies = spark.sql(
        "SELECT year , COUNT(imdb_title_id) as all_movies_2010 FROM
IMDb_movies where year=2010 GROUP BY year")
    count_2010_movies.show()

    # counting movies of year 2011
    count_2011_movies = spark.sql(
        "SELECT year , COUNT(imdb_title_id) as all_movies_2011 FROM
IMDb_movies where year=2011 GROUP BY year")
    count_2011_movies.show()

    # counting movies of year 2012
    count_2012_movies = spark.sql(
        "SELECT year , COUNT(imdb_title_id) as all_movies_2012 FROM
IMDb_movies where year=2012 GROUP BY year")
    count_2012_movies.show()

    # counting movies of year 2013
    count_2013_movies = spark.sql(
        "SELECT year , COUNT(imdb_title_id) as all_movies_2013 FROM
IMDb_movies where year=2013 GROUP BY year")
    count_2013_movies.show()

    # counting movies of year 2014
    count_2014_movies = spark.sql(
        "SELECT year , COUNT(imdb_title_id) as all_movies_2014 FROM
IMDb_movies where year=2014 GROUP BY year")
    count_2014_movies.show()

    # counting movies of year 2015
    count_2015_movies = spark.sql(
        "SELECT year , COUNT(imdb_title_id) as all_movies_2015 FROM
IMDb_movies where year=2015 GROUP BY year")
    count_2015_movies.show()

    # counting movies of year 2016
    count_2016_movies = spark.sql(
        "SELECT year , COUNT(imdb_title_id) as all_movies_2016 FROM
IMDb_movies where year=2016 GROUP BY year")
    count_2016_movies.show()

    # counting movies of year 2017
```

```
    count_2017_movies = spark.sql(
        "SELECT year , COUNT(imdb_title_id) as all_movies_2017 FROM
IMDb_movies where year=2017 GROUP BY year")
    count_2017_movies.show()

    # counting movies of year 2018
    count_2018_movies = spark.sql(
        "SELECT year , COUNT(imdb_title_id) as all_movies_2018 FROM
IMDb_movies where year=2018 GROUP BY year")
    count_2018_movies.show()

    # Calculating the rating average of the movies for the years
2010,2011,2012,2013,2014,2015,2016, 2017,2018
    # 2010
    average_rating2010 = spark.sql(
        "SELECT AVG (weighted_average_vote) as total_avg_2010 FROM
IMDb_movies JOIN IMDb_ratings ON
IMDb_movies.imdb_title_id=IMDb_ratings.imdb_title_id WHERE year=2010")
    average_rating2010.show()

    # 2011
    average_rating2011 = spark.sql(
        "SELECT AVG (weighted_average_vote) as total_avg_2011 FROM
IMDb_movies JOIN IMDb_ratings ON
IMDb_movies.imdb_title_id=IMDb_ratings.imdb_title_id WHERE year=2011")
    average_rating2011.show()

    # 2012
    average_rating2012 = spark.sql(
        "SELECT AVG (weighted_average_vote) as total_avg_2012 FROM
IMDb_movies JOIN IMDb_ratings ON
IMDb_movies.imdb_title_id=IMDb_ratings.imdb_title_id WHERE year=2012")
    average_rating2012.show()

    # 2013
    average_rating2013 = spark.sql(
        "SELECT AVG (weighted_average_vote) as total_avg_2013 FROM
IMDb_movies JOIN IMDb_ratings ON
IMDb_movies.imdb_title_id=IMDb_ratings.imdb_title_id WHERE year=2013")
    average_rating2013.show()

    # 2014
    average_rating2014 = spark.sql(
        "SELECT AVG (weighted_average_vote) as total_avg_2014 FROM
IMDb_movies JOIN IMDb_ratings ON
IMDb_movies.imdb_title_id=IMDb_ratings.imdb_title_id WHERE year=2014")
    average_rating2014.show()

    # 2015
    average_rating2015 = spark.sql(
        "SELECT AVG (weighted_average_vote) as total_avg_2015 FROM
IMDb_movies JOIN IMDb_ratings ON
IMDb_movies.imdb_title_id=IMDb_ratings.imdb_title_id WHERE year=2015")
    average_rating2015.show()

    # 2016
    average_rating2016 = spark.sql(
        "SELECT AVG (weighted_average_vote) as total_avg_2016 FROM
IMDb_movies JOIN IMDb_ratings ON
IMDb_movies.imdb_title_id=IMDb_ratings.imdb_title_id WHERE year=2016")
```

```
    average_rating2016.show()

    # 2017
    average_rating2017 = spark.sql(
        "SELECT AVG (weighted_average_vote) as total_avg_2017 FROM
IMDb_movies JOIN IMDb_ratings ON
IMDb_movies.imdb_title_id=IMDb_ratings.imdb_title_id WHERE year=2017")
    average_rating2017.show()

    # 2018
    average_rating2018 = spark.sql(
        "SELECT AVG (weighted_average_vote) as total_avg_2018 FROM
IMDb_movies JOIN IMDb_ratings ON
IMDb_movies.imdb_title_id=IMDb_ratings.imdb_title_id WHERE year=2018")
    average_rating2018.show()

    # Total Males and females ratting average for years 2010, 2011, 2012,
2013, 2014, 2015,2016,2017,2018
    # 2010
    # Females
    total_female_avg_rate_2010 = spark.sql(
        "SELECT AVG(females_allages_avg_vote) as totalF_avg_2010 FROM
IMDb_movies JOIN IMDb_ratings ON
IMDb_movies.imdb_title_id=IMDb_ratings.imdb_title_id WHERE year=2010")
    total_female_avg_rate_2010.show()

    # Males
    total_male_avg_rate_2010 = spark.sql(
        "SELECT AVG(males_allages_avg_vote) as totalM_avg_2010 FROM
IMDb_movies JOIN IMDb_ratings ON
IMDb_movies.imdb_title_id=IMDb_ratings.imdb_title_id WHERE year=2010")
    total_male_avg_rate_2010.show()

    # 2011
    # Females
    total_female_avg_rate_2011 = spark.sql(
        "SELECT AVG(females_allages_avg_vote) as totalF_avg_2011 FROM
IMDb_movies JOIN IMDb_ratings ON
IMDb_movies.imdb_title_id=IMDb_ratings.imdb_title_id WHERE year=2011")
    total_female_avg_rate_2011.show()

    # Males
    total_male_avg_rate_2011 = spark.sql(
        "SELECT AVG(males_allages_avg_vote) as totalM_avg_2011 FROM
IMDb_movies JOIN IMDb_ratings ON
IMDb_movies.imdb_title_id=IMDb_ratings.imdb_title_id WHERE year=2011")
    total_male_avg_rate_2011.show()

    # 2012
    # Females
    total_female_avg_rate_2012 = spark.sql(
        "SELECT AVG(females_allages_avg_vote) as totalF_avg_2012 FROM
IMDb_movies JOIN IMDb_ratings ON
IMDb_movies.imdb_title_id=IMDb_ratings.imdb_title_id WHERE year=2012")
    total_female_avg_rate_2012.show()

    # Males
    total_male_avg_rate_2012 = spark.sql(
```

```
        "SELECT AVG(males_allages_avg_vote) as totalM_avg_2012 FROM
IMDb_movies JOIN IMDb_ratings ON
IMDb_movies.imdb_title_id=IMDb_ratings.imdb_title_id WHERE year=2012")
    total_male_avg_rate_2012.show()

    # 2013
    # Females
    total_female_avg_rate_2013 = spark.sql(
        "SELECT AVG(females_allages_avg_vote) as totalF_avg_2013 FROM
IMDb_movies JOIN IMDb_ratings ON
IMDb_movies.imdb_title_id=IMDb_ratings.imdb_title_id WHERE year=2013")
    total_female_avg_rate_2013.show()

    # Males
    total_male_avg_rate_2013 = spark.sql(
        "SELECT AVG(males_allages_avg_vote) as totalM_avg_2013 FROM
IMDb_movies JOIN IMDb_ratings ON
IMDb_movies.imdb_title_id=IMDb_ratings.imdb_title_id WHERE year=2013")
    total_male_avg_rate_2013.show()

    # 2014
    # Females
    total_female_avg_rate_2014 = spark.sql(
        "SELECT AVG(females_allages_avg_vote) as totalF_avg_2014 FROM
IMDb_movies JOIN IMDb_ratings ON
IMDb_movies.imdb_title_id=IMDb_ratings.imdb_title_id WHERE year=2014")
    total_female_avg_rate_2014.show()

    # Males
    total_male_avg_rate_2014 = spark.sql(
        "SELECT AVG(males_allages_avg_vote) as totalM_avg_2014 FROM
IMDb_movies JOIN IMDb_ratings ON
IMDb_movies.imdb_title_id=IMDb_ratings.imdb_title_id WHERE year=2014")
    total_male_avg_rate_2014.show()

    # 2015
    # Females
    total_female_avg_rate_2015 = spark.sql(
        "SELECT AVG(females_allages_avg_vote) as totalF_avg_2015 FROM
IMDb_movies JOIN IMDb_ratings ON
IMDb_movies.imdb_title_id=IMDb_ratings.imdb_title_id WHERE year=2015")
    total_female_avg_rate_2015.show()

    # Males
    total_male_avg_rate_2015 = spark.sql(
        "SELECT AVG(males_allages_avg_vote) as totalM_avg_2015 FROM
IMDb_movies JOIN IMDb_ratings ON
IMDb_movies.imdb_title_id=IMDb_ratings.imdb_title_id WHERE year=2015")
    total_male_avg_rate_2015.show()

    # 2016
    # Females
    total_female_avg_rate_2016 = spark.sql(
        "SELECT AVG(females_allages_avg_vote) as totalF_avg_2016 FROM
IMDb_movies JOIN IMDb_ratings ON
IMDb_movies.imdb_title_id=IMDb_ratings.imdb_title_id WHERE year=2016")
    total_female_avg_rate_2016.show()

    # Males
    total_male_avg_rate_2016 = spark.sql(
```

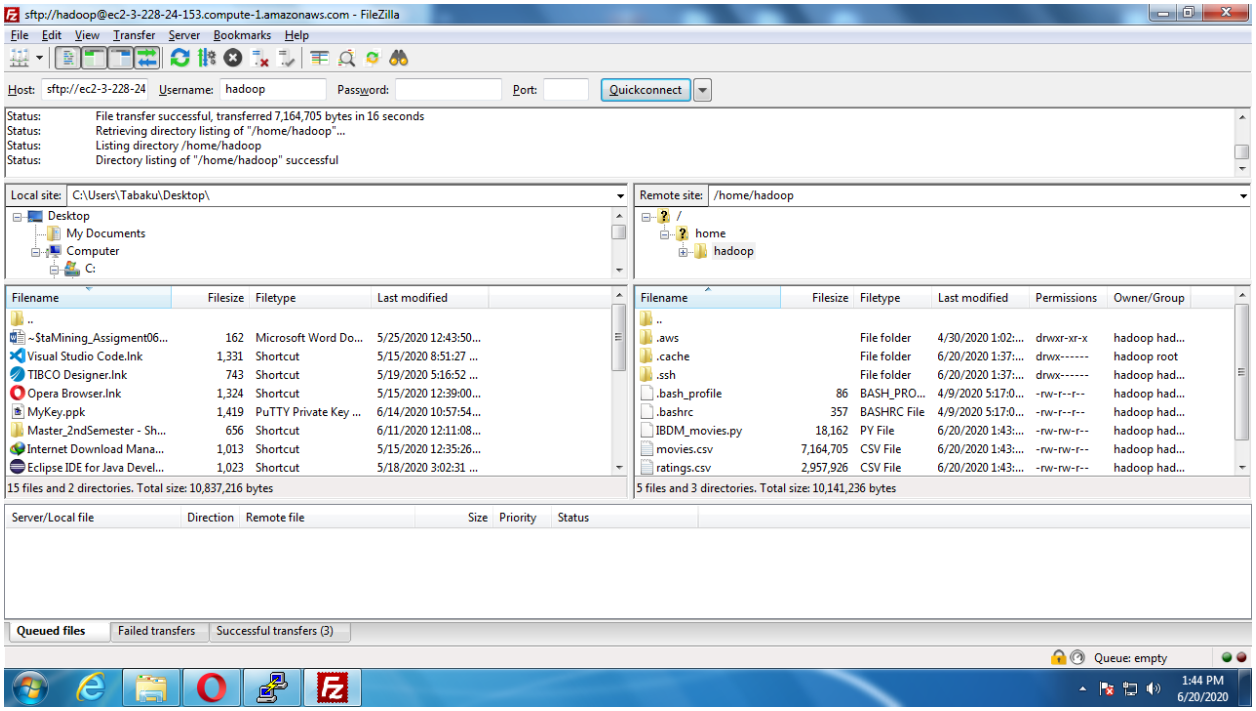
```
        "SELECT AVG(males_allages_avg_vote) as totalM_avg_2016 FROM
IMDb_movies JOIN IMDb_ratings ON
IMDb_movies.imdb_title_id=IMDb_ratings.imdb_title_id WHERE year=2016")
    total_male_avg_rate_2016.show()

    # 2017
    # Females
    total_female_avg_rate_2017 = spark.sql(
        "SELECT AVG(females_allages_avg_vote) as totalF_avg_2017 FROM
IMDb_movies JOIN IMDb_ratings ON
IMDb_movies.imdb_title_id=IMDb_ratings.imdb_title_id WHERE year=2017")
    total_female_avg_rate_2017.show()

    # Males
    total_male_avg_rate_2017 = spark.sql(
        "SELECT AVG(males_allages_avg_vote) as totalM_avg_2017 FROM
IMDb_movies JOIN IMDb_ratings ON
IMDb_movies.imdb_title_id=IMDb_ratings.imdb_title_id WHERE year=2017")
    total_male_avg_rate_2017.show()

    # 2018
    # Females
    total_female_avg_rate_2018 = spark.sql(
        "SELECT AVG(females_allages_avg_vote) as totalF_avg_2018 FROM
IMDb_movies JOIN IMDb_ratings ON
IMDb_movies.imdb_title_id=IMDb_ratings.imdb_title_id WHERE year=2018")
    total_female_avg_rate_2018.show()

    # Males
    total_male_avg_rate_2018 = spark.sql(
        "SELECT AVG(males_allages_avg_vote) as totalM_avg_2018 FROM
IMDb_movies JOIN IMDb_ratings ON
IMDb_movies.imdb_title_id=IMDb_ratings.imdb_title_id WHERE year=2018")
    total_male_avg_rate_2018.show()

    # ============================Saving data==========================
    # TODO , finishing this part, saving the tables

    IMDb_RDD_movies.write.format('csv').option('header',
'true').save(sys.argv[3] + "/output_movies/")
    IMDb_RDD_ratings.write.format('csv').option('header',
'true').save(sys.argv[3] + "/output_ratings/")

    # udemy.write.format('csv').option('header',
'true').save("PROJ/OutputMain/")

    # simple selected and sorted
    selected_all_movies.coalesce(1).write.csv(sys.argv[3] + "/AllMovies")
    # Movies 2010-2018 counted
    count_2010_movies.coalesce(1).write.csv(sys.argv[3] + "/Movies2010Num/")
    count_2011_movies.coalesce(1).write.csv(sys.argv[3] + "/Movies2011Num/")
    count_2012_movies.coalesce(1).write.csv(sys.argv[3] + "/Movies2012Num/")
    count_2013_movies.coalesce(1).write.csv(sys.argv[3] + "/Movies2013Num/")
    count_2014_movies.coalesce(1).write.csv(sys.argv[3] + "/Movies2014Num/")
    count_2015_movies.coalesce(1).write.csv(sys.argv[3] + "/Movies2015Num/")
    count_2016_movies.coalesce(1).write.csv(sys.argv[3] + "/Movies2016Num/")
    count_2017_movies.coalesce(1).write.csv(sys.argv[3] + "/Movies2017Num/")
    count_2018_movies.coalesce(1).write.csv(sys.argv[3] + "/Movies2018Num/")

    # Rating average
```

```
    average_rating2010.coalesce(1).write.csv(sys.argv[3] + "/RatingAVG2010/")
    average_rating2011.coalesce(1).write.csv(sys.argv[3] + "/RatingAVG2011/")
    average_rating2012.coalesce(1).write.csv(sys.argv[3] + "/RatingAVG2012/")
    average_rating2013.coalesce(1).write.csv(sys.argv[3] + "/RatingAVG2013/")
    average_rating2014.coalesce(1).write.csv(sys.argv[3] + "/RatingAVG2016/")
    average_rating2015.coalesce(1).write.csv(sys.argv[3] + "/RatingAVG2015/")
    average_rating2016.coalesce(1).write.csv(sys.argv[3] + "/RatingAVG2016/")
    average_rating2017.coalesce(1).write.csv(sys.argv[3] + "/RatingAVG2017/")
    average_rating2018.coalesce(1).write.csv(sys.argv[3] + "/RatingAVG2018/")

    # Males and females rating average for years 2010 - 2018
    # females
    total_female_avg_rate_2010.coalesce(1).write.csv(sys.argv[3] +
"/FemaleRatingAVG2010/")
    total_female_avg_rate_2011.coalesce(1).write.csv(sys.argv[3] +
"/FemaleRatingAVG2011/")
    total_female_avg_rate_2012.coalesce(1).write.csv(sys.argv[3] +
"/FemaleRatingAVG2012/")
    total_female_avg_rate_2013.coalesce(1).write.csv(sys.argv[3] +
"/FemaleRatingAVG2013/")
    total_female_avg_rate_2014.coalesce(1).write.csv(sys.argv[3] +
"/FemaleRatingAVG2014/")
    total_female_avg_rate_2015.coalesce(1).write.csv(sys.argv[3] +
"/FemaleRatingAVG2015/")
    total_female_avg_rate_2016.coalesce(1).write.csv(sys.argv[3] +
"/FemaleRatingAVG2016/")
    total_female_avg_rate_2017.coalesce(1).write.csv(sys.argv[3] +
"/FemaleRatingAVG2017/")
    total_female_avg_rate_2018.coalesce(1).write.csv(sys.argv[3] +
"/FemaleRatingAVG2018/")

    # Males
    total_male_avg_rate_2010.coalesce(1).write.csv(sys.argv[3] +
"/MaleRatingAVG2010/")
    total_male_avg_rate_2011.coalesce(1).write.csv(sys.argv[3] +
"/MaleRatingAVG2011/")
    total_male_avg_rate_2012.coalesce(1).write.csv(sys.argv[3] +
"/MaleRatingAVG2012/")
    total_male_avg_rate_2013.coalesce(1).write.csv(sys.argv[3] +
"/MaleRatingAVG2013/")
    total_male_avg_rate_2014.coalesce(1).write.csv(sys.argv[3] +
"/MaleRatingAVG2014/")
    total_male_avg_rate_2015.coalesce(1).write.csv(sys.argv[3] +
"/MaleRatingAVG2015/")
    total_male_avg_rate_2016.coalesce(1).write.csv(sys.argv[3] +
"/MaleRatingAVG2016/")
    total_male_avg_rate_2017.coalesce(1).write.csv(sys.argv[3] +
"/MaleRatingAVG2017/")
    total_male_avg_rate_2018.coalesce(1).write.csv(sys.argv[3] +
"/MaleRatingAVG2018/")

    # Movies 2010-2018 counted - END

    spark.stop()
```

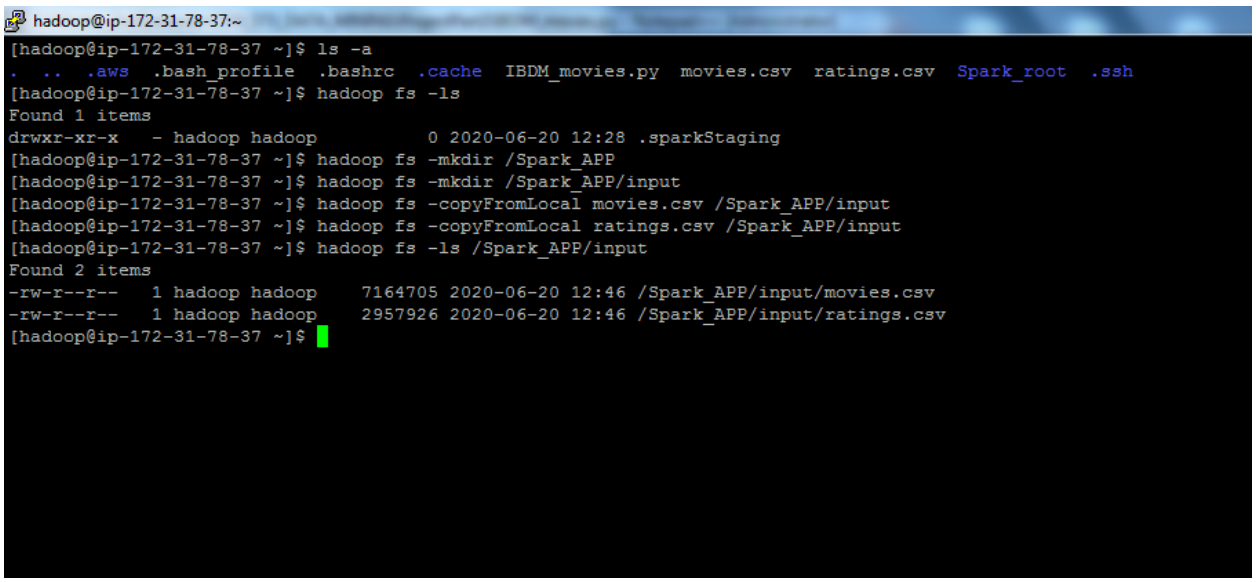
4. Uploading the data into the cluster and processing calculations.

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

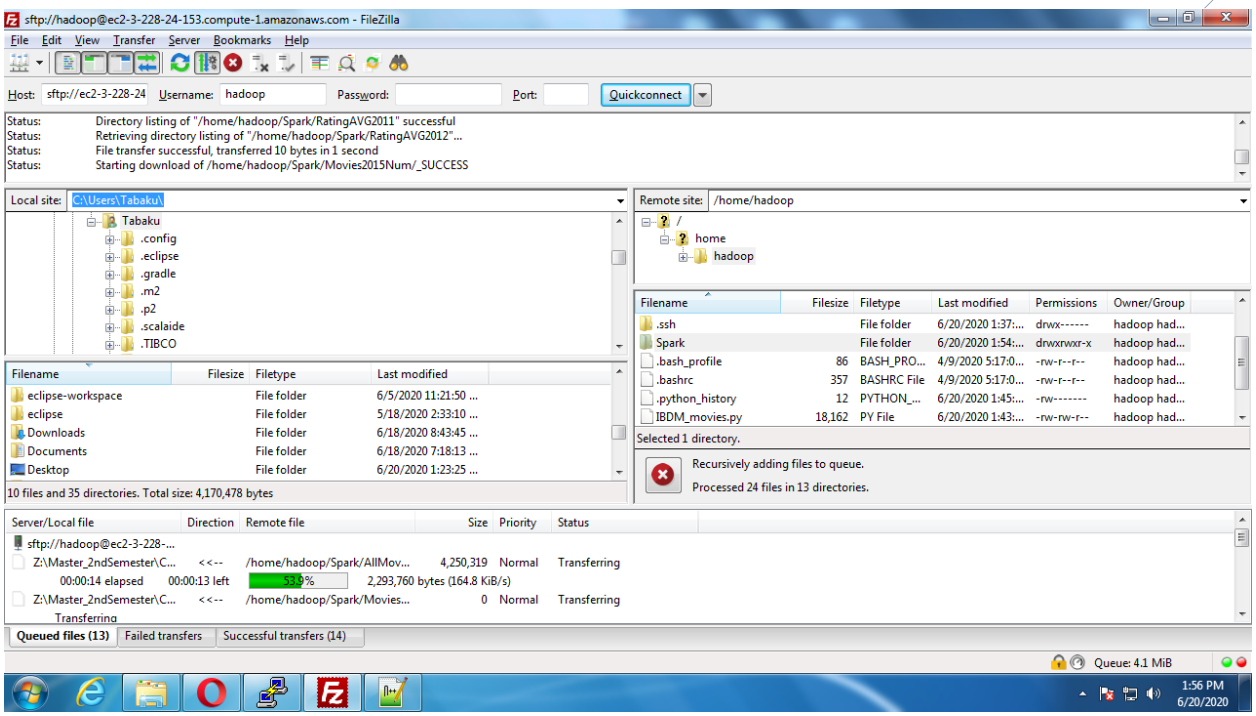
The output files and folders will be stored in the output folder, according to the ordering of the submission, below are the corresponding output screenshots.



Files successfully passed into the cluster

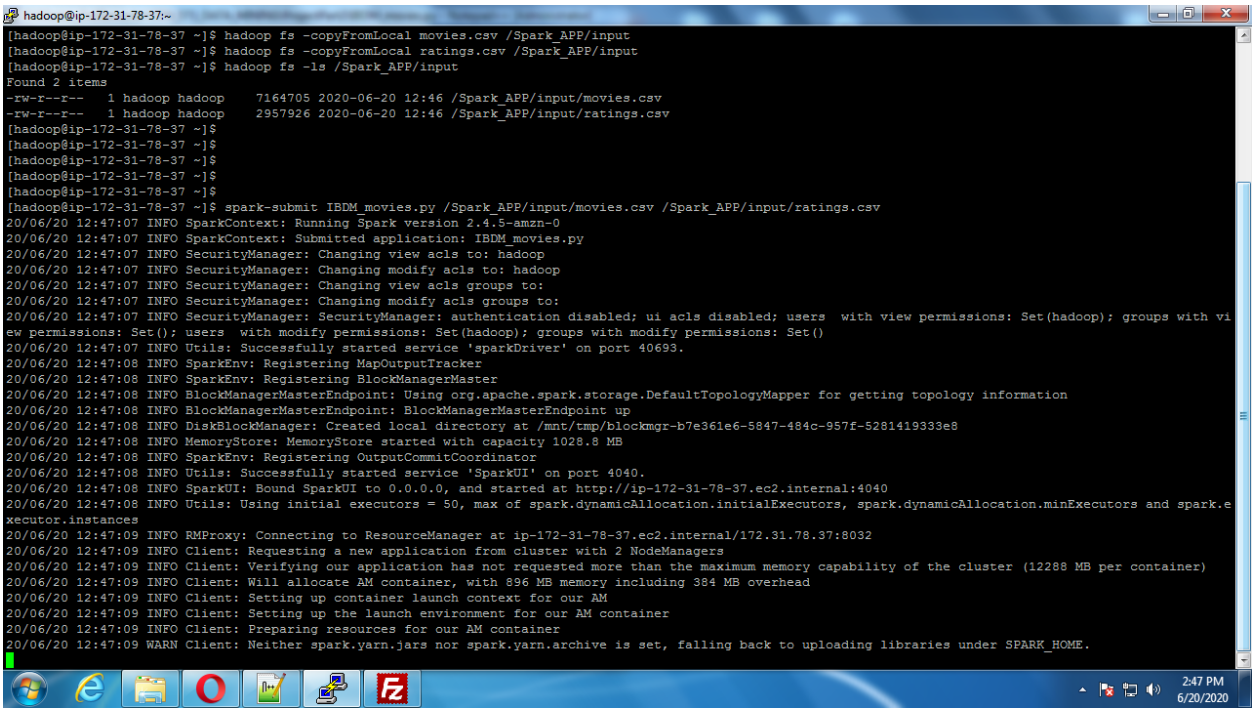


After execution, data files will be transferred again to local directory for further analyze,

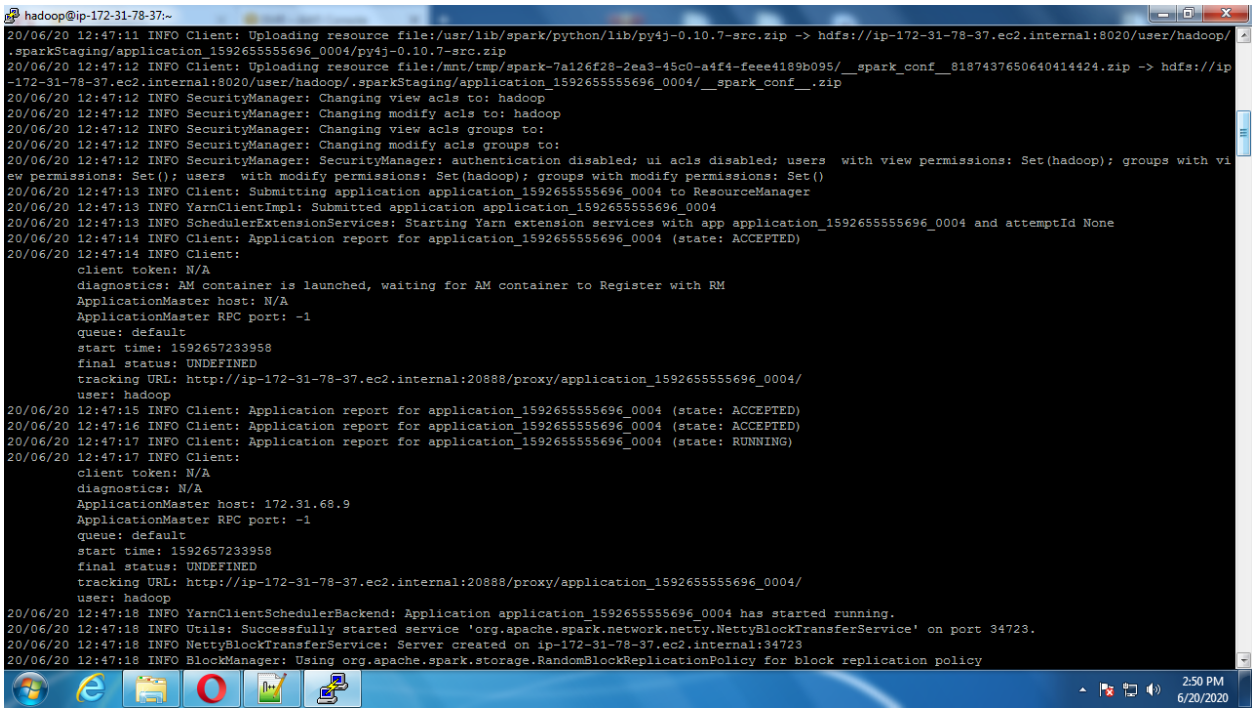


5. Spark used commands and their output

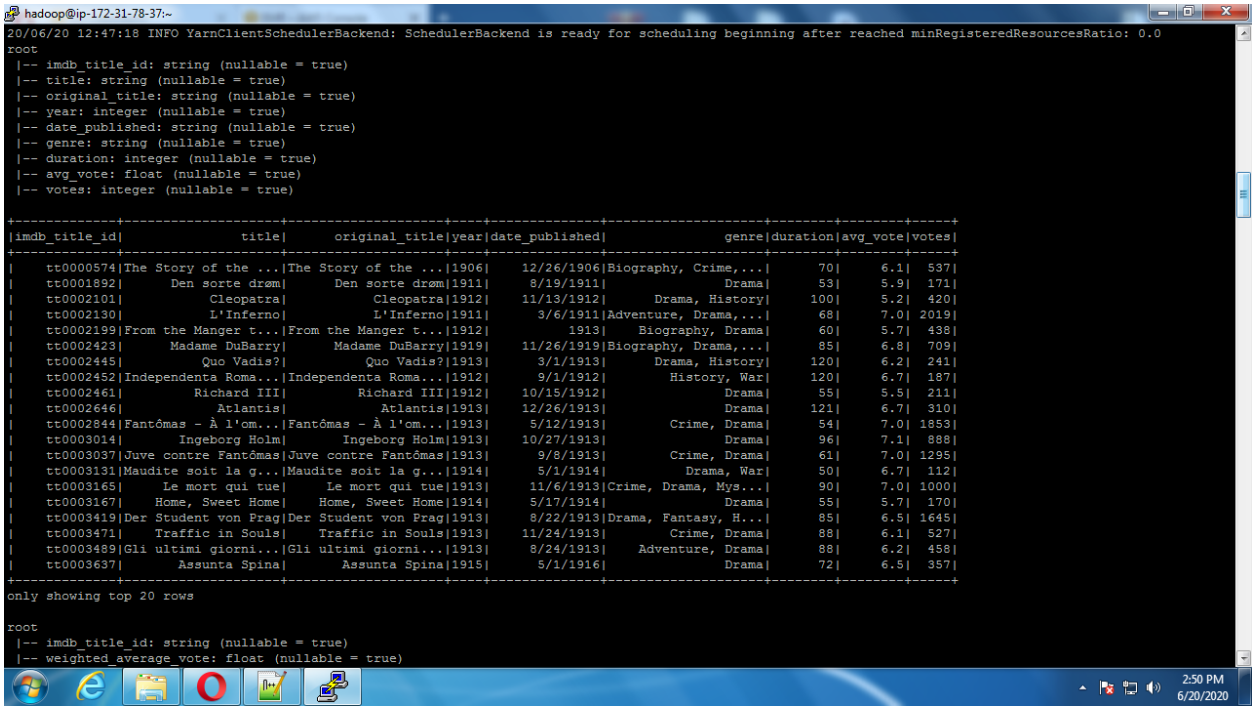
I choose the python do these, I have included the commands in the code part, they are inside the python file, better than running them one by one.



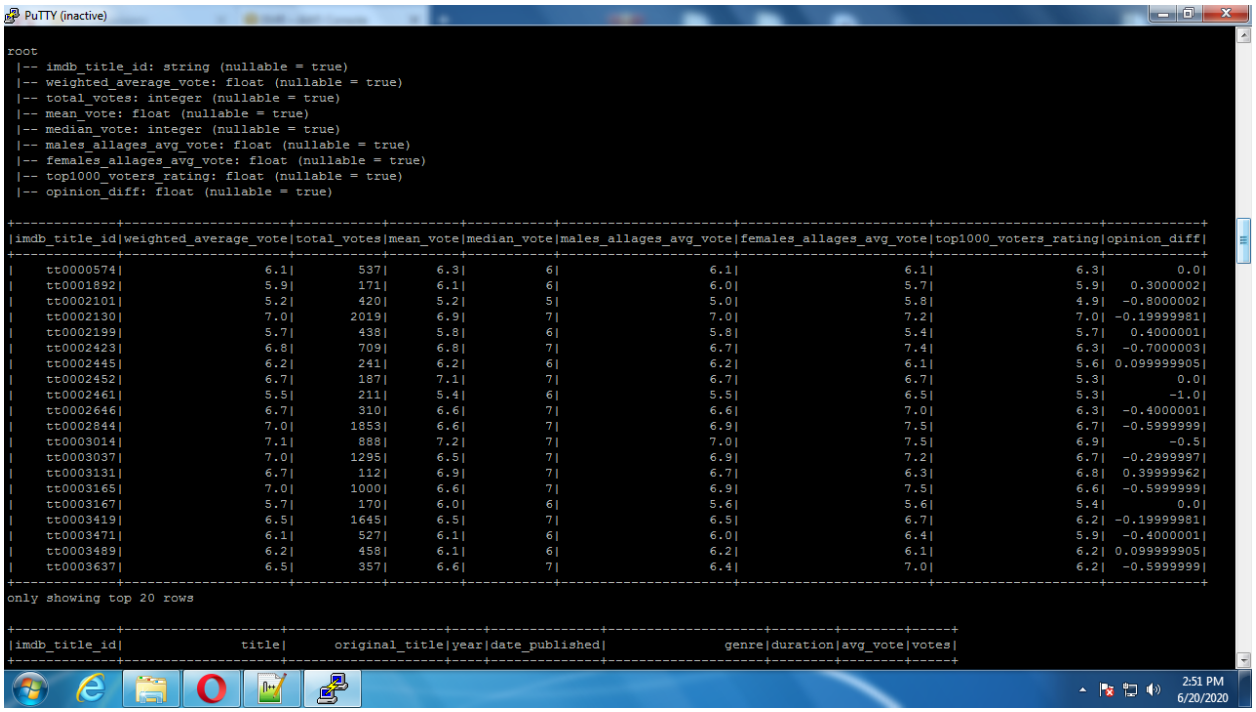
Due to the fact that there are like 60.000 + records to millions, I decided to use the ".show(10)" records or each, then the results, with full selections are saved automatically in CSV files according to their records.



Tables and structures, the movies table,



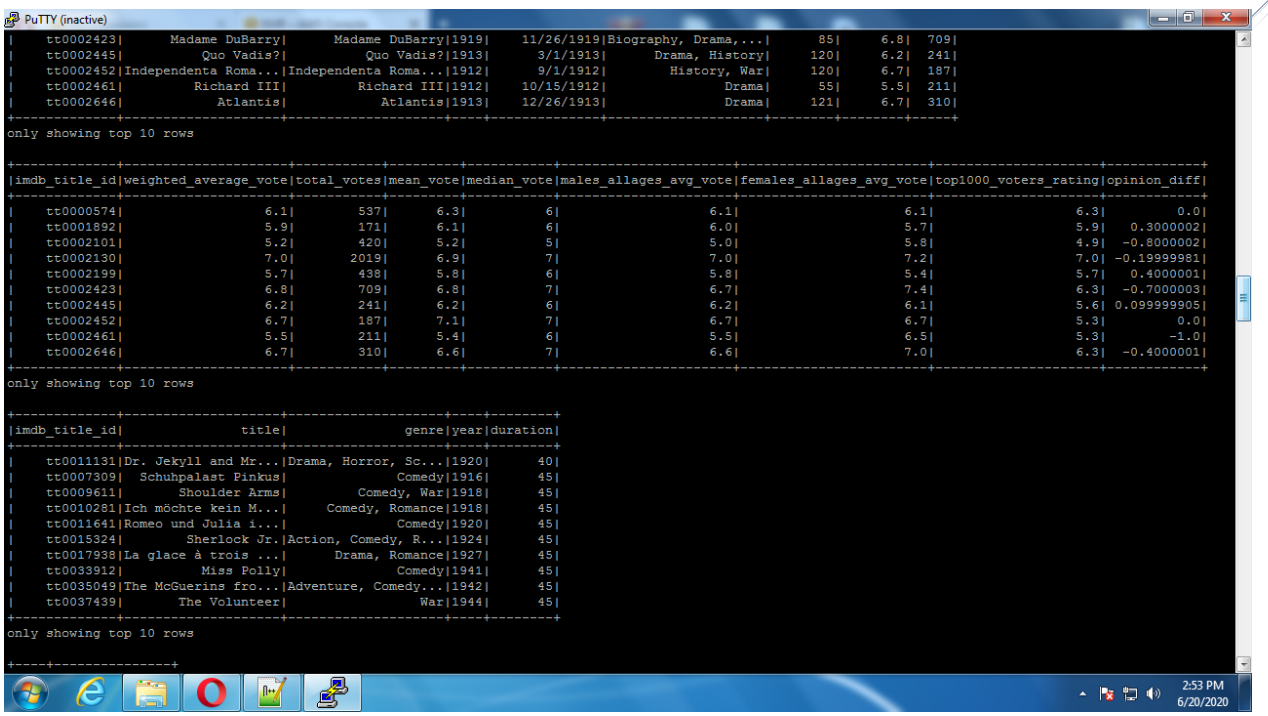
The ratings table, there will be added also an ‘opinion_diff’ column which will be the total male and female opinion difference about a certain movie.



```
PuTTY (inactive)

root
 |-- imdb_title_id: string (nullable = true)
 |-- weighted_average_vote: float (nullable = true)
 |-- total_votes: integer (nullable = true)
 |-- mean_vote: float (nullable = true)
 |-- median_vote: integer (nullable = true)
 |-- males_allages_avg_vote: float (nullable = true)
 |-- females_allages_avg_vote: float (nullable = true)
 |-- top1000_voters_rating: float (nullable = true)
 |-- opinion_diff: float (nullable = true)

+-------------+---------------------+-----------+---------+-----------+----------------------+------------------------+---------------------+------------+
|imdb_title_id|weighted_average_vote|total_votes|mean_vote|median_vote|males_allages_avg_vote|females_allages_avg_vote|top1000_voters_rating|opinion_diff|
+-------------+---------------------+-----------+---------+-----------+----------------------+------------------------+---------------------+------------+
|    tt0000574|                  6.1|        537|      6.3|          6|                   6.1|                     6.1|                  6.3|         0.0|
|    tt0001892|                  5.9|        171|      6.1|          6|                   6.0|                     5.7|                  5.9|   0.3000002|
|    tt0002101|                  5.2|        420|      5.2|          5|                   5.0|                     5.8|                  4.9|  -0.8000002|
|    tt0002130|                  7.0|       2019|      6.9|          7|                   7.0|                     7.2|                  7.0| -0.19999981|
|    tt0002199|                  5.7|        438|      5.8|          6|                   5.8|                     5.4|                  5.7|   0.4000001|
|    tt0002423|                  6.8|        709|      6.8|          7|                   6.7|                     7.4|                  6.3|  -0.7000003|
|    tt0002445|                  6.2|        241|      6.2|          6|                   6.2|                     6.1|                  5.6| 0.099999905|
|    tt0002452|                  6.7|        187|      7.1|          7|                   6.7|                     6.7|                  5.3|         0.0|
|    tt0002461|                  5.5|        211|      5.4|          6|                   5.5|                     6.5|                  5.3|        -1.0|
|    tt0002646|                  6.7|        310|      6.6|          7|                   6.6|                     7.0|                  6.3|  -0.4000001|
|    tt0002844|                  7.0|       1853|      6.6|          7|                   6.9|                     7.5|                  6.7|  -0.5999999|
|    tt0003014|                  7.1|        888|      7.2|          7|                   7.0|                     7.5|                  6.9|        -0.5|
|    tt0003037|                  7.0|       1295|      6.5|          7|                   6.9|                     7.2|                  6.7|  -0.2999997|
|    tt0003131|                  6.7|        112|      6.9|          7|                   6.7|                     6.3|                  6.8|  0.39999962|
|    tt0003165|                  7.0|       1000|      6.6|          7|                   6.9|                     7.5|                  6.6|  -0.5999999|
|    tt0003167|                  5.7|        170|      6.0|          6|                   5.6|                     5.6|                  5.4|         0.0|
|    tt0003419|                  6.5|       1645|      6.5|          7|                   6.5|                     6.7|                  6.2| -0.19999981|
|    tt0003471|                  6.1|        527|      6.1|          6|                   6.0|                     6.4|                  5.9|  -0.4000001|
|    tt0003489|                  6.2|        458|      6.1|          6|                   6.2|                     6.1|                  6.2| 0.099999905|
|    tt0003637|                  6.5|        357|      6.6|          7|                   6.4|                     7.0|                  6.2|  -0.5999999|
+-------------+---------------------+-----------+---------+-----------+----------------------+------------------------+---------------------+------------+
only showing top 20 rows

+-------------+--------------------+--------------------+----+--------------+--------------------+--------+--------+-----+
|imdb_title_id|               title|      original_title|year|date_published|               genre|duration|avg_vote|votes|
+-------------+--------------------+--------------------+----+--------------+--------------------+--------+--------+-----+

2:51 PM
6/20/2020
```

Simple select on movies,

```
+-------------+--------------------+--------------------+----+--------------+--------------------+--------+--------+-----+
|imdb_title_id|               title|      original_title|year|date_published|               genre|duration|avg_vote|votes|
+-------------+--------------------+--------------------+----+--------------+--------------------+--------+--------+-----+
|    tt0000574|The Story of the ...|The Story of the ...|1906|    12/26/1906|Biography, Crime,...|      70|     6.1|  537|
|    tt0001892|      Den sorte drøm|      Den sorte drøm|1911|     8/19/1911|               Drama|      53|     5.9|  171|
|    tt0002101|           Cleopatra|           Cleopatra|1912|    11/13/1912|      Drama, History|     100|     5.2|  420|
|    tt0002130|          L'Inferno|          L'Inferno|1911|     3/6/1911|Adventure, Drama,...|      68|     7.0| 2019|
|    tt0002199|From the Manger t...|From the Manger t...|1912|          1913|    Biography, Drama|      60|     5.7|  438|
|    tt0002423|      Madame DuBarry|      Madame DuBarry|1919|    11/26/1919|Biography, Drama,...|      85|     6.8|  709|
|    tt0002445|          Quo Vadis?|          Quo Vadis?|1913|     3/1/1913|      Drama, History|     120|     6.2|  241|
|    tt0002452|Independenta Roma...|Independenta Roma...|1912|     9/1/1912|       History, War|     120|     6.7|  187|
|    tt0002461|         Richard III|         Richard III|1912|    10/15/1912|               Drama|      55|     5.5|  211|
|    tt0002646|            Atlantis|            Atlantis|1913|    12/26/1913|               Drama|     121|     6.7|  310|
+-------------+--------------------+--------------------+----+--------------+--------------------+--------+--------+-----+
only showing top 10 rows
```



An sorted selection, according to the duration of first 10 rows

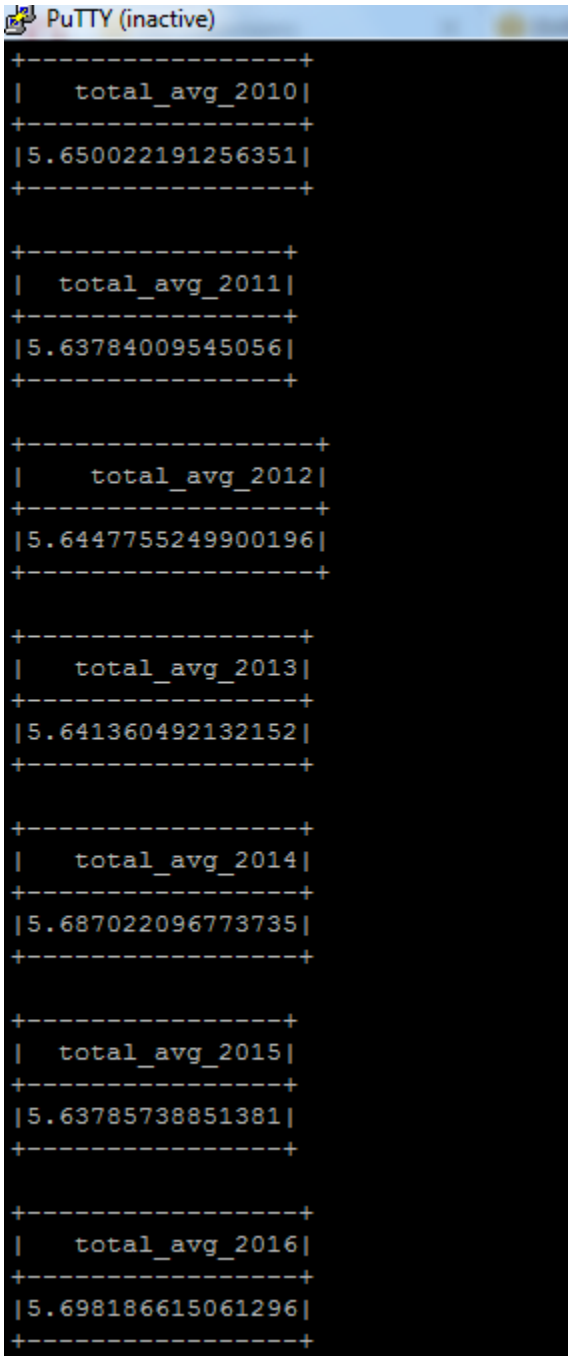
```
+-------------+--------------------+--------------------+----+--------+
|imdb_title_id|               title|               genre|year|duration|
+-------------+--------------------+--------------------+----+--------+
|    tt0011131|Dr. Jekyll and Mr...|Drama, Horror, Sc...|1920|      40|
|    tt0007309|  Schuhpalast Pinkus|              Comedy|1916|      45|
|    tt0009611|       Shoulder Arms|         Comedy, War|1918|      45|
|    tt0010281|Ich möchte kein M...|     Comedy, Romance|1918|      45|
|    tt0011641|Romeo und Julia i...|              Comedy|1920|      45|
|    tt0015324|        Sherlock Jr.|Action, Comedy, R...|1924|      45|
|    tt0017938|La glace à trois ...|      Drama, Romance|1927|      45|
|    tt0033912|          Miss Polly|              Comedy|1941|      45|
|    tt0035049|The McGuerins fro...|Adventure, Comedy...|1942|      45|
|    tt0037439|       The Volunteer|                 War|1944|      45|
+-------------+--------------------+--------------------+----+--------+
only showing top 10 rows

+----+----------------+
```

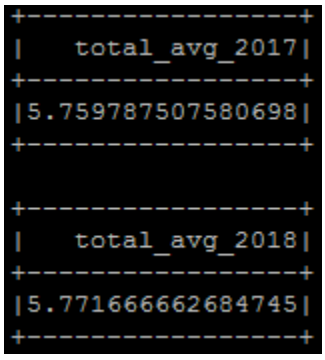
Movies over the years,

```
+----+---------------+
|year|all_movies_2010|
+----+---------------+
|2010|           2253|
+----+---------------+

+----+---------------+
|year|all_movies_2011|
+----+---------------+
|2011|           2389|
+----+---------------+

+----+---------------+
|year|all_movies_2012|
+----+---------------+
|2012|           2517|
+----+---------------+

+----+---------------+
|year|all_movies_2013|
+----+---------------+
|2013|           2749|
+----+---------------+

+----+---------------+
|year|all_movies_2014|
+----+---------------+
|2014|           2851|
+----+---------------+
```

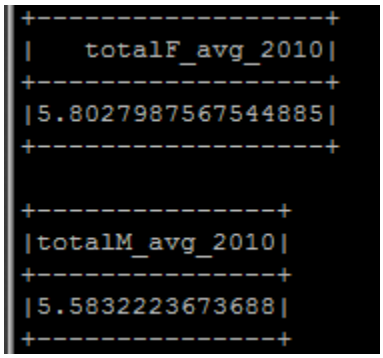
```
+----+---------------+
|year|all_movies_2017|
+----+---------------+
|2017|           3106|
+----+---------------+

+----+---------------+
|year|all_movies_2018|
+----+---------------+
|2018|           2880|
+----+---------------+
```

Average rating for movies over the years 2010-2018



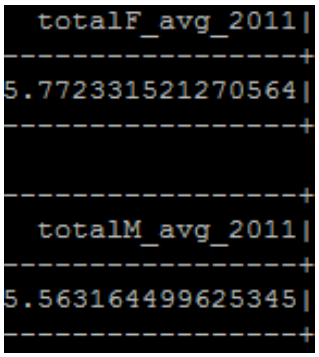


Average ratings for booth males and females, for years 2010 – 2018,



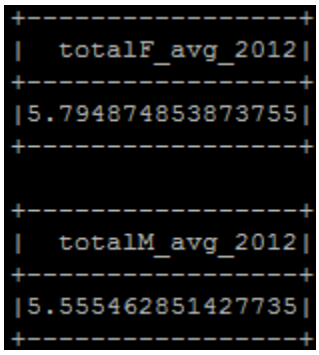
totalF_avg_2010
5.8027987567544885

totalM_avg_2010
5.5832223673688



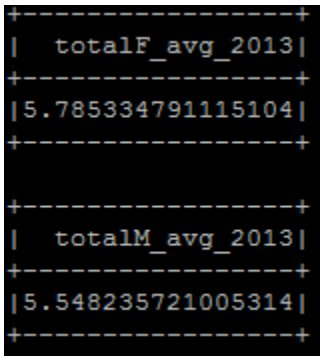
totalF_avg_2011
5.772331521270564

totalM_avg_2011
5.563164499625345



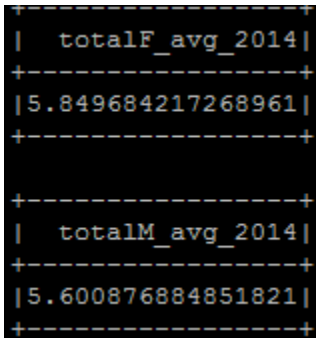
totalF_avg_2012
5.794874853873755

totalM_avg_2012
5.555462851427735



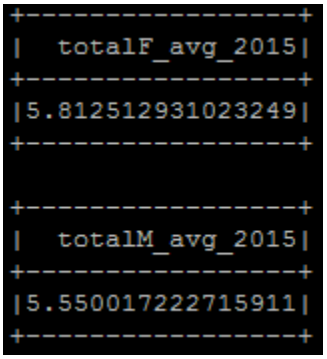
totalF_avg_2013
5.785334791115104

totalM_avg_2013
5.548235721005314



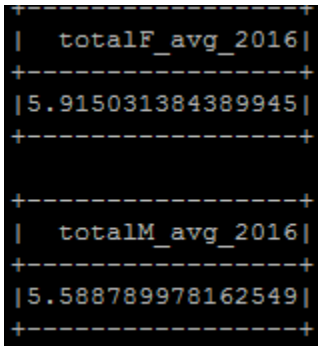
totalF_avg_2014
5.849684217268961

totalM_avg_2014
5.600876884851821



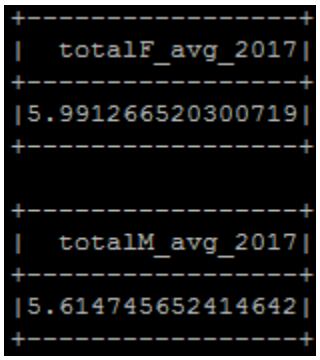
totalF_avg_2015
5.812512931023249

totalM_avg_2015
5.550017222715911



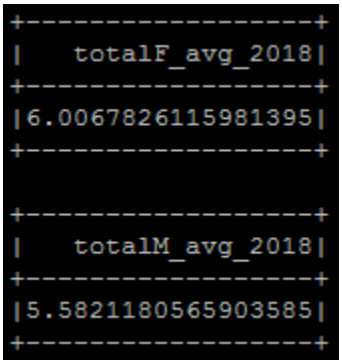
totalF_avg_2016
5.915031384389945

totalM_avg_2016
5.588789978162549



totalF_avg_2017
5.991266520300719

totalM_avg_2017
5.614745652414642



totalF_avg_2018
6.0067826115981395

totalM_avg_2018
5.5821180565903585

Some data folders,

```
Traceback (most recent call last):
  File "/home/hadoop/IBDM_movies.py", line 312, in <module>
    average_rating2016.coalesce(1).write.csv(sys.argv[3] + "/RatingAVG2016/")
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/sql/readwriter.py", line 932, in csv
  File "/usr/lib/spark/python/lib/py4j-0.10.7-src.zip/py4j/java_gateway.py", line 1257, in __call__
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/sql/utils.py", line 69, in deco
pyspark.sql.utils.AnalysisException: 'path hdfs://ip-172-31-66-54.ec2.internal:8020/SPRK_root/RatingAVG2016 already exists.;'
[hadoop@ip-172-31-66-54 ~]$ hadoop fs -ls /SPRK_root
Found 19 items
drwxr-xr-x   - hadoop hadoop          0 2020-06-20 13:39 /SPRK_root/AllMovies
drwxr-xr-x   - hadoop hadoop          0 2020-06-20 13:39 /SPRK_root/Movies2010Num
drwxr-xr-x   - hadoop hadoop          0 2020-06-20 13:39 /SPRK_root/Movies2011Num
drwxr-xr-x   - hadoop hadoop          0 2020-06-20 13:39 /SPRK_root/Movies2012Num
drwxr-xr-x   - hadoop hadoop          0 2020-06-20 13:39 /SPRK_root/Movies2013Num
drwxr-xr-x   - hadoop hadoop          0 2020-06-20 13:39 /SPRK_root/Movies2014Num
drwxr-xr-x   - hadoop hadoop          0 2020-06-20 13:39 /SPRK_root/Movies2015Num
drwxr-xr-x   - hadoop hadoop          0 2020-06-20 13:39 /SPRK_root/Movies2016Num
drwxr-xr-x   - hadoop hadoop          0 2020-06-20 13:39 /SPRK_root/Movies2017Num
drwxr-xr-x   - hadoop hadoop          0 2020-06-20 13:39 /SPRK_root/Movies2018Num
drwxr-xr-x   - hadoop hadoop          0 2020-06-20 13:39 /SPRK_root/RatingAVG2010
drwxr-xr-x   - hadoop hadoop          0 2020-06-20 13:39 /SPRK_root/RatingAVG2011
drwxr-xr-x   - hadoop hadoop          0 2020-06-20 13:39 /SPRK_root/RatingAVG2012
drwxr-xr-x   - hadoop hadoop          0 2020-06-20 13:39 /SPRK_root/RatingAVG2013
drwxr-xr-x   - hadoop hadoop          0 2020-06-20 13:39 /SPRK_root/RatingAVG2015
drwxr-xr-x   - hadoop hadoop          0 2020-06-20 13:39 /SPRK_root/RatingAVG2016
drwxr-xr-x   - hadoop hadoop          0 2020-06-20 13:38 /SPRK_root/input
drwxr-xr-x   - hadoop hadoop          0 2020-06-20 13:39 /SPRK_root/output_movies
drwxr-xr-x   - hadoop hadoop          0 2020-06-20 13:39 /SPRK_root/output_ratings
[hadoop@ip-172-31-66-54 ~]$ hadoop fs -copyToLocal /SPRK_root
[hadoop@ip-172-31-66-54 ~]$
```