Solution – Exercise 04

Spark, PySpark, Jupyter

Solution

Prerequisites:

- Execute all preparation and example tasks of previous HandsOn slides of last lecture
  - Install and setup Spark
  - Install and setup PySpark
  - Install and setup Jupyter

- Start HDFS and YARN
- Start Spark/PySpark/Jupyter

See:

https://github.com/marcelmittelstaedt/BigData/tree/master/solutions/04_spark_pyspark_jupyter/JupyterExercise.html

…for complete solution (Jupyter Notebook).

Solution

Exercise 2:

a) How many **movies** are within the IMDB dataset?

```
In [9]: from pyspark.sql.functions import col
        movie_count = imdb_movie_dataframe.filter(col('titleType') == 'movie').count()

In [11]: print movie_count

         499667
```

b) Who is the **oldest** actor/writer/… within the dataset?

```
In [13]: imdb_people_dataframe = spark.read.format('com.databricks.spark.csv').options(delimiter = '\t',header ='true',nullValu
         e ='null',inferSchema='true').load('hdfs:///user/hadoop/names.basics.tsv')

In [19]: oldest_one = imdb_people_dataframe.filter(col('birthYear') != '\N').sort(col("birthYear").asc()).show(3)

+---------+-----------------+---------+---------+-------------------+-------------------+
|   nconst|      primaryName|birthYear|deathYear|   primaryProfession|      knownForTitles|
+---------+-----------------+---------+---------+-------------------+-------------------+
|nm8572003|   Michael Vignola|     0001|       \N|composer,music_de...|tt6998038,tt40992...|
|nm0784172|Lucio Anneo Seneca|     0004|     0065|             writer|tt0218822,tt09725...|
|nm0194670|       Célina Cély|     0004|       \N|            actress|tt0043347,tt00560...|
+---------+-----------------+---------+---------+-------------------+-------------------+
only showing top 3 rows
```

Solution

c) Create a list (`tconst, original_title, start_year, average_rating, num_votes`) of movies which are:

- equal or newer than year 2000
- have an average rating better than 8
- have been voted more than 100.000 times

save result (DataFrame) back to HDFS as CSV File.

```
In [20]: imdb_ratings_dataframe = spark.read.format('com.databricks.spark.csv').options(delimiter = '\t',header ='true',nullVal
         ue ='null',inferSchema='true').load('hdfs:///user/hadoop/imdb_raw/title_ratings/2018/12/7/title.ratings.tsv')

In [25]: all_imdb_dataframe = imdb_ratings_dataframe.join(imdb_movie_dataframe, imdb_ratings_dataframe.tconst == imdb_movie_dat
         aframe.tconst)

In [34]: good_movies = all_imdb_dataframe.filter(col('numVotes') > 100000).filter(col('startYear') > 2000).filter(col('averageR
         ating') > 8.0).filter(col('titleType') == 'movie')

In [35]: good_movies.sort(col("averageRating").desc(),col("numVotes").desc()).select("originalTitle", "startYear", "averageRati
         ng", "numVotes").show(5)
```

```
+--------------------+---------+-------------+--------+
|       originalTitle|startYear|averageRating|numVotes|
+--------------------+---------+-------------+--------+
|     The Dark Knight|     2008|          9.0| 1971593|
|The Lord of the R...|     2003|          8.9| 1425695|
|           Inception|     2010|          8.8| 1752024|
|The Lord of the R...|     2001|          8.8| 1442541|
|The Lord of the R...|     2002|          8.7| 1288857|
+--------------------+---------+-------------+--------+
only showing top 5 rows
```

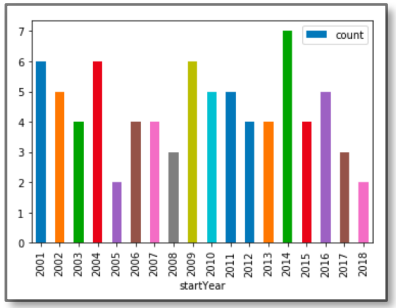
Solution

```
In [36]: good_movies.sort(col("averageRating").desc(),col("numVotes").desc()).select("originalTitle", "startYear", "averageRati
         ng", "numVotes").write.format("csv").save("hdfs:///user/hadoop/good_movies")
         # saved on HDFS as /user/hadoop/good_movies/part-00000-f01c02b6-516a-4a28-9b7e-d271bfc47536-c000.csv
         ...
```

d) How many movies are in list of c)?

```
In [37]: print good_movies.count()
         79
```

e) create following plot with result of c) (plot visualizes the amount of good movies per year since 2001)



Solution

```
In [38]: plot_dataframe = good_movies.groupBy('startYear').count().sort(col("startYear").asc())
```

```
In [39]: plot_dataframe.show()
```

```
+---------+-----+
|startYear|count|
+---------+-----+
|     2001|    6|
|     2002|    5|
|     2003|    4|
|     2004|    6|
|     2005|    2|
|     2006|    4|
|     2007|    4|
|     2008|    3|
|     2009|    6|
|     2010|    5|
|     2011|    5|
|     2012|    4|
|     2013|    4|
|     2014|    7|
|     2015|    4|
|     2016|    5|
|     2017|    3|
|     2018|    2|
+---------+-----+
```

```
In [40]: import matplotlib.pyplot as plt
         import pandas
         pd_df=plot_dataframe.select("startYear", "count").toPandas()
```

```
In [41]: pd_df.plot.bar(x='startYear',y='count')
```

```
Out[41]: <matplotlib.axes._subplots.AxesSubplot at 0x7fe72495ec50>
```