

Napisać kod, który sprawdzi ile razy każdy film został oceniony oraz wyświetlić te informacje w postaci posortowanej wg liczby ocen.

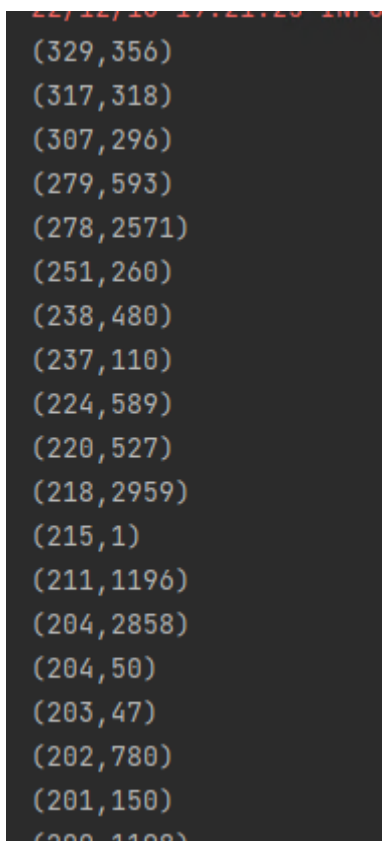
```
public class Main {
    public static void main(String[] args) {

        SparkConf conf = new SparkConf()
            .setAppName("")
            .setMaster("local");

        JavaSparkContext sc = new JavaSparkContext(conf);
// Wczytanie pliku CSV jako kolekcji linii tekstowych
        JavaRDD<String> ratings = sc.textFile("ratings.csv");

// Zmapowanie kolekcji z liniami tekstu na kolekcję z tablic wartości rozbitych wg symbolu przecinka
        JavaRDD<String[]> ratingsArrays = ratings.map(r -> r.split(","));
// Zmapowanie kolekcji z tablicami wartości tekstowych na kolekcję dwójek klucz-wartość (id filmu, 1)
        JavaPairRDD<String, Integer> pairs = ratingsArrays.mapToPair(w -> new Tuple2<>(w[1], 1));
// Zredukowanie kolekcji dwójek wg klucza (id filmu) w taki sposób, aby przy każdej redukcji elementów o takim samym
kluczu liczona była suma ich wartości
        JavaPairRDD<String,Integer> counts = pairs.reduceByKey((a, b) -> a + b);
// Odwrócenie kolejności klucz-wartość.
        JavaPairRDD<Integer, String> invCounts = counts.mapToPair(x -> x.swap());
// Posortowanie kolekcji wynikowej wg klucza
        JavaPairRDD<Integer, String> sorted = invCounts.sortByKey(false);
        sorted.foreach(e -> System.out.println(e));
    }
}
```

Wynik zapytania ile razy każdy film został oceniony w postaci posortowanej wg liczby ocen.



```
(329,356)
(317,318)
(307,296)
(279,593)
(278,2571)
(251,260)
(238,480)
(237,110)
(224,589)
(220,527)
(218,2959)
(215,1)
(211,1196)
(204,2858)
(204,50)
(203,47)
(202,780)
(201,150)
(200,1198)
```

Przystosować program do uruchamiania na klastrze z użyciem skryptu spark-submit.

```
public class Main {
    public static void main(String[] args) {

// Zmiana trybu pracy
        SparkConf conf = new SparkConf();
        JavaSparkContext sc = new JavaSparkContext(conf);

// Załadowniie pliku z klastra z systemem hdfs
        JavaRDD<String> ratings = sc.textFile("hdfs://hadoop1:9000/students/st36148/ratings.csv");

        JavaRDD<String[]> ratingsArrays = ratings.map(r -> r.split(","));
        JavaPairRDD<String, Integer> pairs = ratingsArrays.mapToPair(w -> new Tuple2<>(w[1], 1));
        JavaPairRDD<String,Integer> counts = pairs.reduceByKey((a, b) -> a + b);
        JavaPairRDD<Integer, String> invCounts = counts.mapToPair(x -> x.swap());
        JavaPairRDD<Integer, String> sorted = invCounts.sortByKey(false);

// Zapisanie wyników w klastrze hdfs
        sorted.saveAsTextFile("hdfs:/students/st36148/lab2-ratings-sorted-1part.txt");
    }
}
```

Po napisaniu aplikacji została ona spakowana przy pomocy komendy „package” w środowisku IntelliJ. Plik wynikowy został przesłany na serwer przy pomocy komendy:

```
scp ./target/lab2-1.0-SNAPSHOT.jar st36148@31.193.99.136:/home/st36148
```

Następnie plik .jar oraz pełna wersja pliku ratings.csv zostały przesłane na serwer hadoop:

```
hdfs dfs -put lab2-1.0-SNAPSHOT.jar /students/st36148
```

Aplikacja została uruchomiona za pomocą komendy spark-submit z dołączonym connector’em mysql:

```
spark-submit --class Main --master spark://hadoop1:7077 lab2-1.0-SNAPSHOT.jar
```

Wczytując dane do RDD, zostały one podzielone na sześć partycji, co skutuje utworzeniem sześciu części pliku wynikowego:

```
(8,148448)
st36148@hadoop1:~/lab2-ratings-sorted.txt$ ls -l
total 604
-rw-r--r-- 1 st36148 students 99158 gru 11 12:42 part-00000
-rw-r--r-- 1 st36148 students 97431 gru 11 12:42 part-00001
-rw-r--r-- 1 st36148 students 99310 gru 11 12:42 part-00002
-rw-r--r-- 1 st36148 students 82826 gru 11 12:42 part-00003
-rw-r--r-- 1 st36148 students 111646 gru 11 12:42 part-00004
-rw-r--r-- 1 st36148 students 111356 gru 11 12:42 part-00005
-rw-r--r-- 1 st36148 students 0 gru 11 12:42 _SUCCESS
st36148@hadoop1:~/lab2-ratings-sorted.txt$
```

Zawartość pliku part-00000:

```
(97999,318)
(97040,356)
(92406,296)
(87899,593)
(84545,2571)
(81815,260)
(76451,480)
(71516,527)
(68803,110)
(68469,1)
(66023,1210)
(65822,1196)
(65678,2959)
(64258,589)
(63505,1198)
(62180,50)
(61883,4993)
(60904,858)
(60820,2858)
(58949,780)
(58665,150)
(58031,457)
```

Metoda `coalesce()` zwraca nowy Dataset, który podziela liczbę partycji przekazaną w parametrze.

Po zastosowaniu redukcji liczby partycji do jednej:

```
sorted.coalesce(1).saveAsTextFile("hdfs://students/st36148/lab2-ratings-sorted-1part.txt");
```

tworzony jest jeden plik wyjściowy:

```
st36148@hadoop1:~$ ls -l lab2-ratings-sorted-1part.txt/
total 588
-rw-r--r-- 1 st36148 students 601727 gru 11 12:59 part-00000
-rw-r--r-- 1 st36148 students      0 gru 11 12:59 _SUCCESS
st36148@hadoop1:~$
```

Zawartość pliku part-00000:

```
(97999,318)
(97040,356)
(92406,296)
(87899,593)
(84545,2571)
(81815,260)
(76451,480)
(71516,527)
(68803,110)
(68469,1)
(66023,1210)
(65822,1196)
(65678,2959)
(64258,589)
(63505,1198)
(62180,50)
(61883,4993)
(60904,858)
(60820,2858)
(58949,780)
(58665,150)
(58031,457)
(57492,1270)
(57378,7153)
(56696,5952)
```