**Anna Plust, 36148**

**Duże Zbiory Danych, 2022-23**

**Lab 2, Sprawozdanie**

**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.**

**A picture containing text

Description automatically generated**

**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);

**// Załadownie pliku z klastra z system 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](mailto: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:

Text

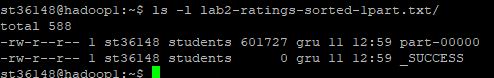
Description automatically generated

Zawartość pliku part-00000:  
Graphical user interface, text, chat or text message

Description automatically generated

Metoda coalesce() zwraca nowy Dataset, który podziada 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:



Zawartość pliku part-00000:  
Graphical user interface, text

Description automatically generated