Anna Plust, 36148

Duże zbiory danych, 2022-23

Lab 5, Sprawozdanie

Napisanie aplikacji wczytującej dane z plików na dysku i uruchamianej lokalnie. (1, 2, 3, 4)

```
public class Main {
    public static void main(String[] args) {
        SparkConf conf = new SparkConf()
                .setAppName("")
                .setMaster("local");

        JavaSparkContext sc = new JavaSparkContext(conf);

        SparkSession spark = SparkSession
                .builder()
                .sparkContext(sc.sc())
                .getOrCreate();

// załadować pliki movies.csv oraz movies.json od oddzielnych obiektów Data Frame (typ Dataset<Row>)
        Dataset<Row> moviesCsv = spark.read().csv("movies.csv");
        Dataset<Row> moviesJson = spark.read().json("movies.json");

// wyświetlić ich zawartość (metoda: show()) oraz opis schematu (metoda: printSchema())
        moviesCsv.show();
        moviesCsv.printSchema();
        moviesJson.show();
        moviesJson.printSchema();

// utworzyć widok SQL dla jednego z utworzonych wcześniej obiektów DataFrame
        moviesJson.createOrReplaceTempView("movies");
// wysłać do widoku zapytanie wg schematu: SELECT * FROM widok WHERE movieId > 500 ORDER BY movieId DESC
        Dataset<Row> sqlDF = spark.sql("SELECT * FROM movies WHERE movieId > 500 ORDER BY movieId DESC");
//wyświetlić rezultat zapytania
        sqlDF.show();

// wczytać dane do kolekcji Dataset zwierającej obiekty typu Movie (czyli Dataset<Movie>)
        Dataset<Movie> df = moviesJson.as(Encoders.bean(Movie.class));

// wczytać plik z danymi o ocenach filmów (ratings.csv)
        Dataset<Row> ratings = spark.read()
                .option("header", true)
                .option("inferSchema", true)
                .csv("ratings.csv");

// wykonać operację grupowania ocen wg identyfikatora filmu oraz policzenia wystąpień w ramach grupy
        Dataset x1 = ratings.groupBy(col("movieId")).count();
// wykonać operację grupowania ocen wg identyfikatora filmu oraz policzenia średniej oceny w ramach grupy
        Dataset x2 = ratings.groupBy(col("movieId")).avg("rating");
// połączyć wyniki liczenia wystąpień i średniej wg identyfikatora filmu i wybranie tylko filmów ocenionych przynajmniej 5 razy
        Dataset x3 = x1.join(x2, "movieId").where(col("count").geq(5));
// połączyć wynik otrzymany w poprzednim podpunkcie ze zbiorem danych o filmach zawierającym tytuły
        Dataset x4_1 = moviesJson.select(col("movieId"), col("title"));
        Dataset x4 = x3.join(x4_1, "movieId");
// wyświetlić tytuły filmów wraz z wartością średniej ocen posortowane wg jej wartości
        Dataset x5 = x4.select(col("movieId"), col("avg(rating)")).orderBy("avg(rating)");
        x5.show();
    }
}
```

Aplikacja została uruchomiona w środowisku IntelliJ, a wyniki widoczne są następnej stronie.

1) Ładowanie obiektów Data Frame z plików tekstowych.
Załadować pliki movies.csv oraz movies.json od oddzielnych obiektów Data Frame (typ Dataset<Row>). Wyświetlić ich zawartość (metoda: show()) oraz opis schematu (metoda: printSchema()).

Plik CSV

```
+-------+--------------------+--------------------+
|    _c0|                 _c1|                 _c2|
+-------+--------------------+--------------------+
|movieId|               title|              genres|
|      1|    Toy Story (1995)|Adventure|Animati...|
|      2|      Jumanji (1995)|Adventure|Childre...|
|      3|Grumpier Old Men ...|      Comedy|Romance|
|      4|Waiting to Exhale...|Comedy|Drama|Romance|
|      5|Father of the Bri...|              Comedy|
|      6|         Heat (1995)|Action|Crime|Thri...|
|      7|      Sabrina (1995)|      Comedy|Romance|
|      8| Tom and Huck (1995)|  Adventure|Children|
|      9| Sudden Death (1995)|              Action|
|     10|    GoldenEye (1995)|Action|Adventure|...|
|     11|American Presiden...|Comedy|Drama|Romance|
|     12|Dracula: Dead and...|       Comedy|Horror|
|     13|        Balto (1995)|Adventure|Animati...|
|     14|        Nixon (1995)|               Drama|
|     15|Cutthroat Island ...|Action|Adventure|...|
|     16|       Casino (1995)|         Crime|Drama|
|     17|Sense and Sensibi...|       Drama|Romance|
|     18|   Four Rooms (1995)|              Comedy|
|     19|Ace Ventura: When...|              Comedy|
+-------+--------------------+--------------------+
only showing top 20 rows

root
 |-- _c0: string (nullable = true)
 |-- _c1: string (nullable = true)
 |-- _c2: string (nullable = true)
```

Plik JSON

```
+--------------------+-------+--------------------+
|              genres|movieId|               title|
+--------------------+-------+--------------------+
|Adventure|Animati...|      1|    Toy Story (1995)|
|Adventure|Childre...|      2|      Jumanji (1995)|
|      Comedy|Romance|      3|Grumpier Old Men ...|
|Comedy|Drama|Romance|      4|Waiting to Exhale...|
|              Comedy|      5|Father of the Bri...|
|Action|Crime|Thri...|      6|         Heat (1995)|
|      Comedy|Romance|      7|      Sabrina (1995)|
|  Adventure|Children|      8| Tom and Huck (1995)|
|              Action|      9| Sudden Death (1995)|
|Action|Adventure|...|     10|    GoldenEye (1995)|
|Comedy|Drama|Romance|     11|American Presiden...|
|       Comedy|Horror|     12|Dracula: Dead and...|
|Adventure|Animati...|     13|        Balto (1995)|
|               Drama|     14|        Nixon (1995)|
|Action|Adventure|...|     15|Cutthroat Island ...|
|         Crime|Drama|     16|       Casino (1995)|
|       Drama|Romance|     17|Sense and Sensibi...|
|              Comedy|     18|   Four Rooms (1995)|
|              Comedy|     19|Ace Ventura: When...|
|Action|Comedy|Cri...|     20|  Money Train (1995)|
+--------------------+-------+--------------------+
only showing top 20 rows

root
 |-- genres: string (nullable = true)
 |-- movieId: long (nullable = true)
 |-- title: string (nullable = true)
```

2) Zadawanie zapytań w języku SQL.

Utworzyć widok SQL dla jednego z utworzonych wcześniej obiektów DataFrame. Wysłać do widoku zapytanie wg schematu: SELECT * FROM widok WHERE movieId > 500 ORDER BY movieId DESC. Wyświetlić rezultat zapytania.

```
+--------------------+-------+--------------------+
|              genres|movieId|               title|
+--------------------+-------+--------------------+
|              Comedy| 193609|Andrew Dice Clay:...|
|    Action|Animation| 193587|Bungo Stray Dogs:...|
|               Drama| 193585|        Flint (2017)|
|Animation|Comedy|...| 193583|No Game No Life: ...|
|Action|Animation|...| 193581|Black Butler: Boo...|
|         Documentary| 193579|Jon Stewart Has L...|
|           Animation| 193573|Love Live! The Sc...|
|        Comedy|Drama| 193571| Silver Spoon (2014)|
|     Animation|Drama| 193567|anohana: The Flow...|
|Action|Animation|...| 193565|Gintama: The Movi...|
|Action|Adventure|...| 191005|      Gintama (2017)|
|         Documentary| 190221|Hommage à Zgougou...|
|           Animation| 190219|        Bunny (1998)|
|               Drama| 190215| Liquid Truth (2017)|
|               Drama| 190213|    John From (2015)|
|              Comedy| 190209|Jeff Ross Roasts ...|
|       Drama|Romance| 190207|         Tilt (2011)|
|     Sci-Fi|Thriller| 190183|The Darkest Minds...|
|  Comedy|Crime|Drama| 189713|BlacKkKlansman (2...|
|       Action|Sci-Fi| 189547| Iron Soldier (2010)|
+--------------------+-------+--------------------+
only showing top 20 rows
```

4) Operacje łączenia i agregacji.
Wyświetlić tytuły filmów wraz z wartością średniej ocen posortowane wg jej wartości.

```
+-------+-----------------+
|movieId|      avg(rating)|
+-------+-----------------+
|   6460|              4.9|
| 177593|             4.75|
|  31364|              4.7|
|   2239|4.666666666666667|
|   4334|              4.6|
|   1041|4.590909090909091|
| 106642|4.571428571428571|
|   3451|4.545454545454546|
|   1178|4.541666666666667|
|   1192|              4.5|
|   2732|              4.5|
| 158966|              4.5|
|   3266|              4.5|
|   3152|              4.5|
|   3201|              4.5|
|   7008|              4.5|
|   1104|            4.475|
|   2360|4.458333333333333|
|   1217|4.433333333333334|
|    318|4.429022082018927|
+-------+-----------------+
only showing top 20 rows
```

Dostosowanie aplikacji i następnie uruchomienie jej w środowisku serwera. Wynik uzyskany w pkt. 4 zapisać w formacie CSV systemie plików hdfs klastra. (5, 6)

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

        SparkConf conf = new SparkConf();

        JavaSparkContext sc = new JavaSparkContext(conf);

        SparkSession spark = SparkSession
                .builder()
                .sparkContext(sc.sc())
                .getOrCreate();

        Properties prop = new Properties();

        prop.setProperty("user", "bigdata");
        prop.setProperty("password", "1234");

// Wczytanie danych o filmach z serwera MySql
        Dataset<Row> movies = spark.read().jdbc(
                "jdbc:mysql://zecer.wi.zut.edu.pl:3306/bigdata",
                "movies",
                prop
        );

        Dataset<Row> ratings = spark.read()
                .option("header", true)
                .option("inferSchema", true)
                .csv("hdfs:///students/st36148/ratings.csv");

        Dataset x1 = ratings.groupBy(col("movieId")).count();
        Dataset x2 = ratings.groupBy(col("movieId")).avg("rating");
        Dataset x3 = x1.join(x2, "movieId").where(col("count").geq(5));
        Dataset x4_1 = movies.select(col("movieId"), col("title"));
        Dataset x4 = x3.join(x4_1, "movieId");
        Dataset x5 = x4.select(col("movieId"), col("avg(rating)")).orderBy(desc("avg(rating)"));

// Wynik uzyskany w pkt. 4 zapisać w formacie CSV systemie plików hdfs klastra.
        x5.write().csv("hdfs:/students/st36148/lab2-jdbc.csv");
    }
}
```

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

```
scp ./target/lab5-1.0-SNAPSHOT.jar st36148@31.193.99.136:/home/st36148
```

Natępnie plik został przesłany na serwer hadoop:

```
hdfs dfs -put lab5-1.0-SNAPSHOT.jar /students/st36148
```

Dodatkowo na serwer konieczne było wysłanie paczki *mysql-connector-java-8.0.29.jar*.

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

```
spark-submit --class Main --master spark://hadoop1:7077 --driver-class-path mysql-connector-java-8.0.29.jar --jars mysql-connector-java-8.0.29.jar lab5-1.0-SNAPSHOT.jar
```

Zapisany plik na serwerze hdfs:

```
st36148@hadoop1:~/lab2-jdbc.csv$ hdfs dfs -ls /students/st36148/lab2-jdbc.csv
Found 2 items
-rw-r--r--   3 st36148 supergroup          0 2022-12-15 15:13 /students/st36148/lab2-jdbc.csv/_SUCCESS
-rw-r--r--   3 st36148 supergroup     224867 2022-12-15 15:13 /students/st36148/lab2-jdbc.csv/part-00000-4ccf73ac-df78-4c62-9859-5c22f5ebf2f4-c000.csv
st36148@hadoop1:~/lab2-jdbc.csv$
```

Zawartość pliku:

```
171011,4.4865181711606095
159817,4.458092485549133
318,4.424188001918387
170705,4.399898373983739
174053,4.350558659217877
171495,4.343949044585988
172591,4.339667458432304
858,4.332892749244713
50,4.291958829205532
102742,4.285714285714286
174551,4.275
176601,4.263888888888889
1221,4.2630353697749195
172577,4.261904761904762
527,4.257501817775044
2019,4.2541157909178215
150696,4.25
163809,4.244031830238727
185135,4.23943661971831
1203,4.237075455914338
179135,4.236389684813753
904,4.230798598634567
2959,4.230663235786717
1193,4.222920272160452
142115,4.2137767220902616
912,4.210098086509085
750,4.208876000542667
5618,4.2076678004047015
1212,4.20375939849624
147250,4.20353982300885
166024,4.202861952861953
1178,4.201752440106477
908,4.201091113037271
155168,4.2
174737,4.2
44555,4.199844881075491
3435,4.199261675824176
922,4.195425943852856
100044,4.190812720848056
6016,4.184896558122275
3030,4.178287597042514
```