# SparkSQL 开发

## 一、数据说明

本文使用MovieLens的名称为ml-25m.zip的数据集，使用的文件时movies.csv和rating.csv，上述文件的下载地址为：  
<http://files.grouplens.org/datasets/movielens/ml-25m.zip>

movies.csv是电影数据，对应的为维表数据，包括6万多部电影，其数据格式为[movieId,title,genres],分别对应[电影id，电影名称，电影所属分类]

| movieId | title | genres |
| --- | --- | --- |
| 电影ID | 电影名称 | 电影所属分类 |

样例数据如下  
1,"Toy Story (1995)","Adventure|Animation|Children|Comedy|Fantasy"  
2,"Jumanji (1995)","Adventure|Children|Fantasy"

rating.csv为定影评分数据，对应为事实表数据，大小为646MB，其数据格式为:[userId,movieId,rating,timestamp],分别对应[用户id，电影id，评分，时间戳]

| userId | movieId | rating | timestamp |
| --- | --- | --- | --- |
| 用户ID | 电影ID | 评分 | 时间戳 |

样例数据如下  
1,296,5,1147880044

## 二、需求分析

1. 查找电影评分个数超过5000,且平均评分较高的前十部电影名称及其对应的平均评分
2. 查找每个电影类别及其对应的平均评分
3. 查找被评分次数较多的前十部电影

## 三、分析过程

### 3.1 准备

#### 3.1.1 下载数据并解压

前往网址下载数据，使用如下命令解压缩

unzip movielens.zip

#### 3.1.2 上传文件至HDFS

首先打开hadoop

cd /usr/local/hadoop  
./sbin/start-dfs.sh

hadoop@rorschach-virtual-machine:~/Downloads$ cd /usr/local/hadoop  
hadoop@rorschach-virtual-machine:/usr/local/hadoop$ ./sbin/start-dfs.sh  
Starting namenodes on [localhost]  
Starting datanodes  
Starting secondary namenodes [rorschach-virtual-machine]

可以输入jps查看是否打开成功

hadoop@rorschach-virtual-machine:/usr/local/hadoop$ jps  
8512 NameNode  
9016 Jps  
8893 SecondaryNameNode  
8655 DataNode

将文件上传

hadoop@rorschach-virtual-machine:/usr/local/hadoop$ ./bin/hdfs dfs -mkdir -p /sparksql/input  
hadoop@rorschach-virtual-machine:/usr/local/hadoop$ ./bin/hdfs dfs -put ~/Downloads/movielens/movie.csv /sparksql/input  
hadoop@rorschach-virtual-machine:/usr/local/hadoop$ ./bin/hdfs dfs -put ~/Downloads/movielens/rating.csv /sparksql/input

查看文件是否传输成功

hadoop@rorschach-virtual-machine:/usr/local/hadoop$ ./bin/hdfs dfs -ls /sparksql/input  
Found 2 items  
-rw-r--r-- 1 hadoop supergroup 1493648 2023-10-13 13:37 /sparksql/input/movie.csv  
-rw-r--r-- 1 hadoop supergroup 690353377 2023-10-13 13:38 /sparksql/input/rating.csv

文件上传成功！

#### 3.1.3 查看数据

hadoop@rorschach-virtual-machine:/usr/local/hadoop$ ./bin/hdfs dfs -cat /sparksql/input/movie.csv | head -5  
"movieId","title","genres"  
1,"Toy Story (1995)","Adventure|Animation|Children|Comedy|Fantasy"  
2,"Jumanji (1995)","Adventure|Children|Fantasy"  
3,"Grumpier Old Men (1995)","Comedy|Romance"  
4,"Waiting to Exhale (1995)","Comedy|Drama|Romance"  
cat: Unable to write to output stream.  
  
hadoop@rorschach-virtual-machine:/usr/local/hadoop$ ./bin/hdfs dfs -cat /sparksql/input/rating.csv | head -5  
"userId","movieId","rating","timestamp"  
1,2,3.5,2005-04-02 23:53:47  
1,29,3.5,2005-04-02 23:31:16  
1,32,3.5,2005-04-02 23:33:39  
1,47,3.5,2005-04-02 23:32:07  
cat: Unable to write to output stream.

### 3.2 编写代码

本文使用IntelliJ IDEA进行编写

#### 3.2.1 打开IDEA

hadoop@rorschach-virtual-machine:~/Desktop$ cd /usr/local/idea  
hadoop@rorschach-virtual-machine:/usr/local/idea$ ./bin/idea.sh

#### 3.2.2 创建MAVEN项目

首先创建应用程序根目录

hadoop@rorschach-virtual-machine:~/Desktop$ cd ~  
hadoop@rorschach-virtual-machine:~$ mkdir ./SparkSQLTest  
hadoop@rorschach-virtual-machine:~$ mkdir -p ./SparkSQLTest/src/main/scala/demos  
hadoop@rorschach-virtual-machine:~$ mkdir -p ./SparkSQLTest/src/main/scala/metrics

分别编写scala文件

hadoop@rorschach-virtual-machine:~$ vim ./SparkSQLTest/src/main/scala/demos/SparkSQLTest.scala  
hadoop@rorschach-virtual-machine:~$ vim ./SparkSQLTest/src/main/scala/demos/Entry.scala  
hadoop@rorschach-virtual-machine:~$ vim ./SparkSQLTest/src/main/scala/demos/SchemaLoader.scala  
hadoop@rorschach-virtual-machine:~$ vim ./SparkSQLTest/src/main/scala/metrics/BestFilmByOverallRating.scala  
hadoop@rorschach-virtual-machine:~$ vim ./SparkSQLTest/src/main/scala/metrics/GenresByAverageRating.scala  
hadoop@rorschach-virtual-machine:~$ vim ./SparkSQLTest/src/main/scala/metrics/MostRateFilms.scala

以及添加依赖文件

hadoop@rorschach-virtual-machine:~/SparkSQLTest$ vim pom.xml

项目文件结构如下

hadoop@rorschach-virtual-machine:~/SparkSQLTest$ cd ~/SparkSQLTest  
hadoop@rorschach-virtual-machine:~/SparkSQLTest$ find .  
.  
./src  
./src/main  
./src/main/scala  
./src/main/scala/demos  
./src/main/scala/demos/SparkSQLTest.scala  
./src/main/scala/demos/Entry.scala  
./src/main/scala/demos/SchemaLoader.scala  
./src/main/scala/metrics  
./src/main/scala/metrics/MostRateFilms.scala  
./src/main/scala/metrics/BestFilmByOverallRating.scala  
./src/main/scala/metrics/GenresByAverageRating.scala  
./pom.xml

##### 自此，项目框架搭建完成 😊

#### 3.2.3 编写项目代码

###### 1.添加依赖 pom.xml

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <groupId>SparkSQLTest</groupId>  
 <artifactId>SparkSQL-project</artifactId>  
 <modelVersion>4.0.0</modelVersion>  
 <name>SparkSQL Project</name>  
 <packaging>jar</packaging>  
 <version>1.0</version>  
  
   
<properties>  
 <maven.compiler.source>1.8</maven.compiler.source>  
 <maven.compiler.target>1.8</maven.compiler.target>  
 <spark.version>3.5.0</spark.version>  
 <scala.version>2.12.18</scala.version>  
</properties>  
  
<dependencies>  
  
 <dependency>  
 <groupId>org.apache.spark</groupId>  
 <artifactId>spark-core\_2.12</artifactId>  
 <version>3.5.0</version>  
 </dependency>  
  
  
 <dependency>  
 <groupId>org.apache.spark</groupId>  
 <artifactId>spark-sql\_2.12</artifactId>  
 <version>3.5.0</version>  
 </dependency>  
  
</dependencies>  
  
<build>  
 <plugins>  
  
 <plugin>  
 <groupId>org.scala-tools</groupId>  
 <artifactId>maven-scala-plugin</artifactId>  
 <version>2.15.2</version>  
 <executions>  
 <execution>  
 <goals>  
 <goal>compile</goal>  
 <goal>testCompile</goal>  
 </goals>  
 </execution>  
 </executions>  
 </plugin>  
  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>   
 <artifactId>maven-compiler-plugin</artifactId>  
 <version>3.6.0</version>  
 <configuration>  
 <source>1.8</source>  
 <target>1.8</target>  
 </configuration>  
 </plugin>  
  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-surefire-plugin</artifactId>  
 <version>2.19</version>  
 <configuration>  
 <skip>true</skip>  
 </configuration>  
 </plugin>  
  
 </plugins>  
  
</build>  
</project>

###### 2.SparkSQLtest主类

package demos  
  
import metrics.\_  
import org.apache.spark.sql.SparkSession  
import org.apache.spark.sql.types.StructType  
  
object SparkSQLtest {  
 //文件路径  
 private val MOVIES\_CSV\_FILE\_PATH = "hdfs://localhost:9000/sparksql/input/movie.csv"  
 private val RATINGS\_CSV\_FILE\_PATH = "hdfs://localhost:9000/sparksql/input/rating.csv"  
  
 def main(args: Array[String]): Unit = {  
 // 创建spark session  
 val spark = SparkSession  
 .builder  
 .master("local[4]")  
 .getOrCreate  
 // schema信息  
 val schemaLoader = new SchemaLoader  
 // 读取Movie数据集  
 val movieDF = readCsvIntoDataSet(spark, MOVIES\_CSV\_FILE\_PATH, schemaLoader.getMovieSchema)  
 // 读取Rating数据集  
 val ratingDF = readCsvIntoDataSet(spark, RATINGS\_CSV\_FILE\_PATH, schemaLoader.getRatingSchema)  
  
 // 需求1：查找电影评分个数超过5000,且平均评分较高的前十部电影名称及其对应的平均评分  
 val bestFilmsByOverallRating = new BestFilmsByOverallRating  
 bestFilmsByOverallRating.run(movieDF, ratingDF, spark)  
  
 // 需求2：查找每个电影类别及其对应的平均评分  
 val genresByAverageRating = new GenresByAverageRating  
 genresByAverageRating.run(movieDF, ratingDF, spark)  
  
 // 需求3：查找被评分次数较多的前十部电影  
 val mostRatedFilms = new MostRatedFilms  
 mostRatedFilms.run(movieDF, ratingDF, spark)  
  
 spark.close()  
  
 }  
  
 /\*\*  
 \* 读取数据文件，转成DataFrame  
 \*  
 \* @param spark  
 \* @param path  
 \* @param schema  
 \* @return  
 \*/  
 def readCsvIntoDataSet(spark: SparkSession, path: String, schema: StructType) = {  
  
 val dataSet = spark.read  
 .format("csv")  
 .option("header", "true")  
 .schema(schema)  
 .load(path)  
 dataSet  
 }  
  
}

###### 3.SchemaLoader Schema信息

package demos  
  
import metrics.\_  
import org.apache.spark.sql.types.{DataTypes, StructField, StructType}  
  
  
  
class SchemaLoader {  
 // movies数据集schema信息  
 private val movieSchema = new StructType()  
 .add("movieId", DataTypes.StringType, false)  
 .add("title", DataTypes.StringType, false)  
 .add("genres", DataTypes.StringType, false)  
 // ratings数据集schema信息  
 private val ratingSchema = new StructType()  
 .add("userId", DataTypes.StringType, false)  
 .add("movieId", DataTypes.StringType, false)  
 .add("rating", DataTypes.StringType, false)  
 .add("timestamp", DataTypes.StringType, false)  
  
 def getMovieSchema: StructType = movieSchema  
  
 def getRatingSchema: StructType = ratingSchema  
  
}

###### 4.Entry 封装样例类

package demos  
import metrics.\_  
  
class Entry {  
  
}  
  
case class Movies(  
 movieId: String, // 电影的id  
 title: String, // 电影的标题  
 genres: String // 电影类别  
 )  
  
case class Ratings(  
 userId: String, // 用户的id  
 movieId: String, // 电影的id  
 rating: String, // 用户评分  
 timestamp: String // 时间戳  
 )  
  
// 需求1结果表  
case class tenGreatestMoviesByAverageRating(  
 movieId: String, // 电影的id  
 title: String, // 电影的标题  
 avgRating: String // 电影平均评分  
)  
  
// 需求2结果表  
case class topGenresByAverageRating(  
 genres: String, //电影类别  
 avgRating: String // 平均评分  
 )  
  
// 需求3结果表  
case class tenMostRatedFilms(  
 movieId: String, // 电影的id  
 title: String, // 电影的标题  
 ratingCnt: String // 电影被评分的次数  
 )

###### 5.BestFilmByOverallRating

package metrics  
  
import demos.\_  
import org.apache.spark.sql.{DataFrame, SparkSession}  
  
  
/\*\*  
 \* 需求1：查找电影评分个数超过5000,且平均评分较高的前十部电影名称及其对应的平均评分  
 \*/  
class BestFilmsByOverallRating extends Serializable {  
  
 def run(moviesDataset: DataFrame, ratingsDataset: DataFrame, spark: SparkSession) = {  
 import spark.implicits.\_  
  
 // 将moviesDataset注册成表  
 moviesDataset.createOrReplaceTempView("movies")  
 // 将ratingsDataset注册成表  
 ratingsDataset.createOrReplaceTempView("ratings")  
  
 // 查询SQL语句  
 val ressql1 =  
 """  
 |WITH ratings\_filter\_cnt AS (  
 |SELECT  
 | movieId,  
 | count( \* ) AS rating\_cnt,  
 | avg( rating ) AS avg\_rating  
 |FROM  
 | ratings  
 |GROUP BY  
 | movieId  
 |HAVING  
 | count( \* ) >= 5000  
 |),  
 |ratings\_filter\_score AS (  
 |SELECT  
 | movieId, -- 电影id  
 | avg\_rating -- 电影平均评分  
 |FROM ratings\_filter\_cnt  
 |ORDER BY avg\_rating DESC -- 平均评分降序排序  
 |LIMIT 10 -- 平均分较高的前十部电影  
 |)  
 |SELECT  
 | m.movieId,  
 | m.title,  
 | r.avg\_rating AS avgRating  
 |FROM  
 | ratings\_filter\_score r  
 |JOIN movies m ON m.movieId = r.movieId  
 """.stripMargin  
  
 val resultDS = spark.sql(ressql1).as[tenGreatestMoviesByAverageRating]  
 // 打印数据  
 resultDS.show(10)  
 resultDS.printSchema()  
  
 }  
  
}

###### GenresByAverageRating

package metrics  
  
import demos.\_  
import org.apache.spark.sql.{DataFrame, SparkSession}  
  
/\*\*  
\* 需求2：查找每个电影类别及其对应的平均评分  
\*/  
class GenresByAverageRating extends Serializable {  
 def run(moviesDataset: DataFrame, ratingsDataset: DataFrame, spark: SparkSession) = {  
 import spark.implicits.\_  
 // 将moviesDataset注册成表  
 moviesDataset.createOrReplaceTempView("movies")  
 // 将ratingsDataset注册成表  
 ratingsDataset.createOrReplaceTempView("ratings")  
  
 val ressql2 =  
 """  
 |WITH explode\_movies AS (  
 |SELECT  
 | movieId,  
 | title,  
 | category  
 |FROM  
 | movies lateral VIEW explode ( split ( genres, "\\|" ) ) temp AS category  
 |)  
 |SELECT  
 | m.category AS genres,  
 | avg( r.rating ) AS avgRating  
 |FROM  
 | explode\_movies m  
 | JOIN ratings r ON m.movieId = r.movieId  
 |GROUP BY  
 | m.category  
 | """.stripMargin  
  
 val resultDS = spark.sql(ressql2).as[topGenresByAverageRating]  
  
 // 打印数据  
 resultDS.show(10)  
 resultDS.printSchema()  
  
 }  
}

###### MostRatedFilms

package metrics  
  
import demos.\_  
import org.apache.spark.sql.{DataFrame, SparkSession}  
  
/\*\*  
 \* 需求3：查找被评分次数较多的前十部电影.  
 \*/  
class MostRatedFilms extends Serializable {  
 def run(moviesDataset: DataFrame, ratingsDataset: DataFrame, spark: SparkSession) = {  
  
 import spark.implicits.\_  
  
 // 将moviesDataset注册成表  
 moviesDataset.createOrReplaceTempView("movies")  
 // 将ratingsDataset注册成表  
 ratingsDataset.createOrReplaceTempView("ratings")  
  
 val ressql3 =  
 """  
 |WITH rating\_group AS (  
 | SELECT  
 | movieId,  
 | count( \* ) AS ratingCnt  
 | FROM ratings  
 | GROUP BY movieId  
 |),  
 |rating\_filter AS (  
 | SELECT  
 | movieId,  
 | ratingCnt  
 | FROM rating\_group  
 | ORDER BY ratingCnt DESC  
 | LIMIT 10  
 |)  
 |SELECT  
 | m.movieId,  
 | m.title,  
 | r.ratingCnt  
 |FROM  
 | rating\_filter r  
 |JOIN movies m ON r.movieId = m.movieId  
 |  
 """.stripMargin  
  
 val resultDS = spark.sql(ressql3).as[tenMostRatedFilms]  
 // 打印数据  
 resultDS.show(10)  
 resultDS.printSchema()  
 }  
}

#### 3.2.4 应用程序打包

通过如下代码将整个应用程序打包成JAR包

hadoop@rorschach-virtual-machine:~/SparkSQLTest$ cd ~/SparkSQLTest  
hadoop@rorschach-virtual-machine:~/SparkSQLTest$ /usr/local/maven/bin/mvn package

出现如下信息表明成功

[INFO] Building jar: /home/hadoop/SparkSQLTest/target/SparkSQL-project-1.0.jar  
[INFO] ------------------------------------------------------------------------  
[INFO] BUILD SUCCESS  
[INFO] ------------------------------------------------------------------------  
[INFO] Total time: 5.828 s  
[INFO] Finished at: 2023-10-13T20:39:15+08:00  
[INFO] ------------------------------------------------------------------------

### 3.3 运行程序并查看结果

如下代码运行应用程序包

hadoop@rorschach-virtual-machine:~/SparkSQLTest$ /usr/local/spark/bin/spark-submit --master yarn --class demos.SparkSQLtest ~/SparkSQLTest/target/SparkSQL-project-1.0.jar

查看结果

+-------+--------------------+-----------------+  
|movieId| title| avgRating|  
+-------+--------------------+-----------------+  
| 50|Usual Suspects, T...|4.334372207803259|  
| 318|Shawshank Redempt...|4.446990499637029|  
| 527|Schindler's List ...|4.310175010988133|  
| 858|Godfather, The (1...|4.364732196832306|  
| 904| Rear Window (1954)|4.271333600779414|  
| 912| Casablanca (1942)|4.258326830670664|  
| 922|Sunset Blvd. (a.k...|4.256934865900383|  
| 1193|One Flew Over the...| 4.24807897901911|  
| 1221|Godfather: Part I...|4.275640557704942|  
| 2019|Seven Samurai (Sh...| 4.2741796572216|  
+-------+--------------------+-----------------+  
  
root  
 |-- movieId: string (nullable = true)  
 |-- title: string (nullable = true)  
 |-- avgRating: double (nullable = true)  
  
+-----------+------------------+  
| genres| avgRating|  
+-----------+------------------+  
| Crime|3.6745276025631113|  
| Romance| 3.541802581902903|  
| Thriller| 3.50711121809216|  
| Adventure|3.5018926565473865|  
| Drama|3.6742955093068264|  
| War|3.8095307347384844|  
|Documentary|3.7397176834178865|  
| Fantasy|3.5059453358738244|  
| Mystery| 3.663508921312903|  
| Musical| 3.558090628821412|  
+-----------+------------------+  
only showing top 10 rows  
  
root  
 |-- genres: string (nullable = false)  
 |-- avgRating: double (nullable = true)  
  
+-------+--------------------+---------+  
|movieId| title|ratingCnt|  
+-------+--------------------+---------+  
| 110| Braveheart (1995)| 53769|  
| 260|Star Wars: Episod...| 54502|  
| 296| Pulp Fiction (1994)| 67310|  
| 318|Shawshank Redempt...| 63366|  
| 356| Forrest Gump (1994)| 66172|  
| 480|Jurassic Park (1993)| 59715|  
| 527|Schindler's List ...| 50054|  
| 589|Terminator 2: Jud...| 52244|  
| 593|Silence of the La...| 63299|  
| 2571| Matrix, The (1999)| 51334|  
+-------+--------------------+---------+  
  
root  
 |-- movieId: string (nullable = true)  
 |-- title: string (nullable = true)  
 |-- ratingCnt: long (nullable = false)

## 四、结果说明

### 4.1 需求一：平均评分最高的前十部电影

| movieId | title | avgRating |
| --- | --- | --- |
| 50 | Usual Suspects, T... | 4.334372207803259 |
| 318 | Shawshank Redempt... | 4.446990499637029 |
| 527 | Schindler's List ... | 4.310175010988133 |
| 858 | Godfather, The (1... | 4.364732196832306 |
| 904 | Rear Window (1954) | 4.271333600779414 |
| 912 | Casablanca (1942) | 4.258326830670664 |
| 922 | Sunset Blvd. (a.k... | 4.256934865900383 |
| 1193 | One Flew Over the... | 4.24807897901911 |
| 1221 | Godfather: Part I... | 4.275640557704942 |
| 2019 | Seven Samurai (Sh... | 4.2741796572216 |

对应的中文名称为

| movieId | title | avgRating | 中文名称 |
| --- | --- | --- | --- |
| 50 | Usual Suspects, T... | 4.334372207803259 | 非常嫌疑犯 |
| 318 | Shawshank Redempt... | 4.446990499637029 | 肖申克的救赎 |
| 527 | Schindler's List ... | 4.310175010988133 | 辛德勒的名单 |
| 858 | Godfather, The (1... | 4.364732196832306 | 教父1 |
| 904 | Rear Window (1954) | 4.271333600779414 | 后窗 |
| 912 | Casablanca (1942) | 4.258326830670664 | 卡萨布兰卡 |
| 922 | Sunset Blvd. (a.k... | 4.256934865900383 | 日落大道 |
| 1193 | One Flew Over the... | 4.24807897901911 | 飞跃疯人院 |
| 1221 | Godfather: Part I... | 4.275640557704942 | 教父2 |
| 2019 | Seven Samurai (Sh... | 4.2741796572216 | 七武士 |

### 4.2 需求二：top10电影分类

| genres | avgRating | 中文 |
| --- | --- | --- |
| Crime | 3.6745276025631113 | 犯罪 |
| Romance | 3.541802581902903 | 浪漫 |
| Thriller | 3.50711121809216 | 惊悚 |
| Adventure | 3.5018926565473865 | 冒险 |
| Drama | 3.6742955093068264 | 戏剧 |
| War | 3.8095307347384844 | 战争 |
| Documentary | 3.7397176834178865 | 记录 |
| Fantasy | 3.5059453358738244 | 魔幻 |
| Mystery | 3.663508921312903 | 推理 |
| Musical | 3.558090628821412 | 音乐 |

### 4.3 需求三：评分次数最多

| movieId | title | ratingCnt | 中文 |
| --- | --- | --- | --- |
| 110 | Braveheart (1995) | 53769 | 勇敢的心 |
| 260 | Star Wars: Episod... | 54502 | 星球大战 |
| 296 | Pulp Fiction (1994) | 67310 | 低俗小说 |
| 318 | Shawshank Redempt... | 63366 | 肖申克的救赎 |
| 356 | Forrest Gump (1994) | 66172 | 阿甘正传 |
| 480 | Jurassic Park (1993) | 59715 | 朱罗纪公园 |
| 527 | Schindler's List ... | 50054 | 辛德勒的名单 |
| 589 | Terminator 2: Jud... | 52244 | 终结者 |
| 593 | Silence of the La... | 63299 | 沉默的羔羊 |
| 2571 | Matrix, The (1999) | 51334 | 黑客帝国 |