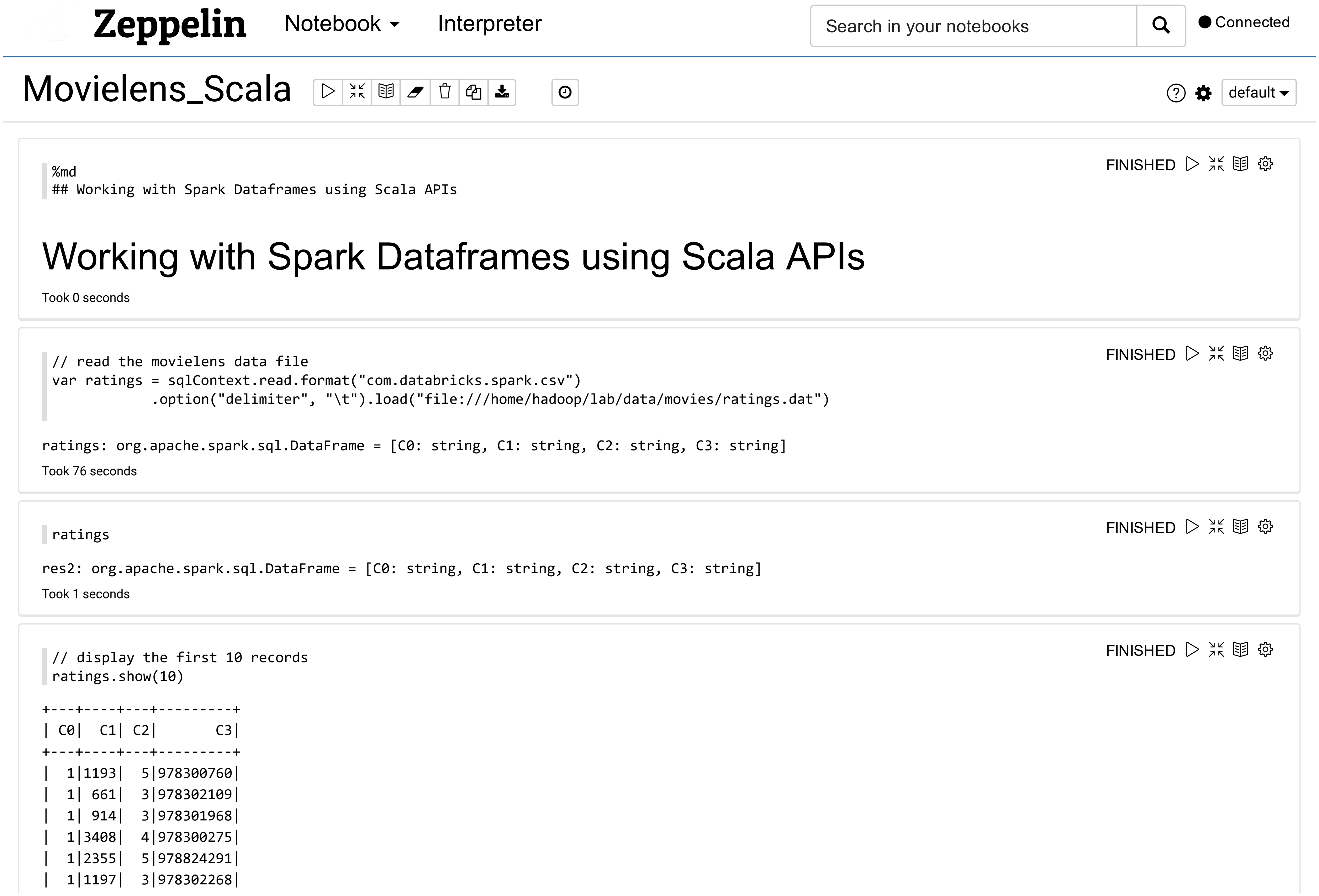

Movielens_Scala

```
%md
## Working with Spark Dataframes using Scala APIs
```

Working with Spark Dataframes using Scala APIs

Took 0 seconds

FINISHED

```
// read the movielens data file
var ratings = sqlContext.read.format("com.databricks.spark.csv")
            .option("delimiter", "\t").load("file:///home/hadoop/lab/data/movies/ratings.dat")
```

```
ratings: org.apache.spark.sql.DataFrame = [C0: string, C1: string, C2: string, C3: string]
```

Took 76 seconds

FINISHED

```
ratings
```

```
res2: org.apache.spark.sql.DataFrame = [C0: string, C1: string, C2: string, C3: string]
```

Took 1 seconds

FINISHED

```
// display the first 10 records
ratings.show(10)
```

```
+---+----+---+---------+
| C0|  C1| C2|       C3|
+---+----+---+---------+
|  1|1193|  5|978300760|
|  1| 661|  3|978302109|
|  1| 914|  3|978301968|
|  1|3408|  4|978300275|
|  1|2355|  5|978824291|
|  1|1197|  3|978302268|
```

FINISHED

Movielens_Scala

```
|   1|1287|  5|978302039|
|   1|2804|  5|978300719|
|   1| 594|  4|978302268|
|   1| 919|  4|978301368|
+----+----+---+---------+
only showing top 10 rows
```

Took 0 seconds

FINISHED

```
// describe the schema of the records
ratings.printSchema()
```

```
root
 |-- C0: string (nullable = true)
 |-- C1: string (nullable = true)
 |-- C2: string (nullable = true)
 |-- C3: string (nullable = true)
```

Took 1 seconds

FINISHED

```
ratings.schema
```

```
res13: org.apache.spark.sql.types.StructType = StructType(StructField(C0,StringType,true), StructField(C1,StringType,true), StructField(C2,StringTyp
e,true), StructField(C3,StringType,true))
```

Took 2 seconds

FINISHED

```
// Applying the schema manually

import org.apache.spark.sql.SQLContext
import org.apache.spark.sql.types.{StructType, StructField, StringType, IntegerType, LongType};

val ratingSchema = StructType(Array(
    StructField("userid", IntegerType, true),
    StructField("movieid", IntegerType, true),
    StructField("rating", IntegerType, true),
    StructField("timestamp", LongType, true)))
```

```
import org.apache.spark.sql.SQLContext
import org.apache.spark.sql.types.{StructType, StructField, StringType, IntegerType, LongType}
ratingSchema: org.apache.spark.sql.types.StructType = StructType(StructField(userid,IntegerType,true), StructField(movieid,IntegerType,true), StructF
ield(rating,IntegerType,true), StructField(timestamp,LongType,true))
```

Took 2 seconds

FINISHED

```
// read the movielens data file with the custom schema
var ratings_df = sqlContext.read.format("com.databricks.spark.csv")
            .option("delimiter", "\t").schema(ratingSchema).load("file:///home/hadoop/lab/data/movies/ratings.dat")
```

```
ratings_df: org.apache.spark.sql.DataFrame = [userid: int, movieid: int, rating: int, timestamp: bigint]
```

Took 0 seconds

FINISHED

```
ratings_df
```

```
res21: org.apache.spark.sql.DataFrame = [userid: int, movieid: int, rating: int, timestamp: bigint]
```

Took 1 seconds

FINISHED

```
ratings_df.show( 5 )
```

```
+------+-------+------+---------+
|userid|movieid|rating|timestamp|
+------+-------+------+---------+
|     1|   1193|     5|978300760|
|     1|    661|     3|978302109|
|     1|    914|     3|978301968|
|     1|   3408|     4|978300275|
|     1|   2355|     5|978824291|
+------+-------+------+---------+
only showing top 5 rows
```

Took 1 seconds

FINISHED

```
ratings_df.printSchema()
```

```
root
 |-- userid: integer (nullable = true)
 |-- movieid: integer (nullable = true)
 |-- rating: integer (nullable = true)
 |-- timestamp: long (nullable = true)
```

Took 0 seconds

FINISHED

```
// Returns a list of columns
ratings_df.columns
```

```
res34: Array[String] = Array(userid, movieid, rating, timestamp)
```

Took 0 seconds

FINISHED

```
// How many records in the dataframe?
ratings_df.count()
```

```
res37: Long = 1000209
```

Took 1 seconds

FINISHED

```
// We donot need the timestamp column.. let's drop it
ratings_df = ratings_df.drop( "timestamp" )
```

```
ratings_df: org.apache.spark.sql.DataFrame = [userid: int, movieid: int, rating: int]
```

Took 0 seconds

FINISHED

```
ratings_df.show(5)
```

```
+------+-------+------+
|userid|movieid|rating|
+------+-------+------+
|     1|   1193|     5|
|     1|    661|     3|
|     1|    914|     3|
|     1|   3408|     4|
|     1|   2355|     5|
+------+-------+------+
only showing top 5 rows
```

Took 1 seconds

FINISHED

```
// Lets's group by movieid and find which movie has been rated by how many users
var movie_counts = ratings_df.groupBy("movieid").count()
```

```
movie_counts: org.apache.spark.sql.DataFrame = [movieid: int, count: bigint]
```

Took 0 seconds (outdated)

FINISHED

```
import org.apache.spark.sql.functions._
movie_counts = movie_counts.sort(desc("count"))
```

```
import org.apache.spark.sql.functions._
movie_counts: org.apache.spark.sql.DataFrame = [movieid: int, count: bigint]
```

Took 1 seconds

FINISHED

```
movie_counts.show( 10 )
```

```
+-------+-----+
|movieid|count|
+-------+-----+
|   2858| 3428|
|    260| 2991|
|   1196| 2990|
|   1210| 2883|
|    480| 2672|
|   2028| 2653|
|    589| 2649|
|   2571| 2590|
|   1270| 2583|
|    593| 2578|
+-------+-----+
only showing top 10 rows
```

Took 3 seconds

FINISHED

```
// average rating for each movie
var avg_ratings = ratings_df.groupBy("movieid").agg( avg( "rating") )
```

```
avg_ratings: org.apache.spark.sql.DataFrame = [movieid: int, avg(rating): double]
```

Took 0 seconds (outdated)

FINISHED

```
avg_ratings.printSchema()
```

```
root
 |-- movieid: integer (nullable = true)
 |-- avg(rating): double (nullable = true)
```

Took 0 seconds

```
avg_ratings = avg_ratings.sort( desc( "avg(rating)" ) )
```

FINISHED

```
avg_ratings: org.apache.spark.sql.DataFrame = [movieid: int, avg(rating): double]
```

Took 0 seconds

```
avg_ratings.show( 10 )
```

FINISHED

```
+-------+-----------+
|movieid|avg(rating)|
+-------+-----------+
|   3607|        5.0|
|    989|        5.0|
|    787|        5.0|
|   3172|        5.0|
|   3656|        5.0|
|   3881|        5.0|
|   3382|        5.0|
|   3233|        5.0|
|   3280|        5.0|
|   1830|        5.0|
+-------+-----------+
only showing top 10 rows
```

Took 3 seconds

```
var avg_ratings_count = avg_ratings.join( movie_counts, "movieid" )
// An Alternate way of joining two dataframes, in case the column names are different
//var avg_ratings_count = avg_ratings.join( movie_counts, avg_ratings("movieid") === movie_counts("movieid") )
```

FINISHED

```
avg_ratings_count: org.apache.spark.sql.DataFrame = [movieid: int, avg(rating): double, count: bigint]
```

Took 1 seconds

```
avg_ratings_count.show(10)
```

FINISHED

```
+-------+------------------+-----+
|movieid|       avg(rating)|count|
+-------+------------------+-----+
|     31|3.1134751773049647|  141|
|    231|3.1924242424242424|  660|
|    431|3.6910569105691056|  369|
|    631|              2.08|   75|
|    831|              3.85|   20|
|   1031| 3.479623824451411|  319|
|   1231|              4.08|  750|
|   1431|2.5474137931034484|  232|
|   1631|3.5918367346938775|   49|
|   1831|2.5847076461769114|  667|
+-------+------------------+-----+
only showing top 10 rows
```

Took 75 seconds

FINISHED

```
avg_ratings_count.printSchema()
```

```
root
 |-- movieid: integer (nullable = true)
 |-- avg(rating): double (nullable = true)
 |-- count: long (nullable = false)
```

Took 1 seconds

FINISHED

```
// rename the column avg(rating) to mean_rating
avg_ratings_count = avg_ratings_count.withColumnRenamed( "avg(rating)", "mean_rating" )
```

```
avg_ratings_count: org.apache.spark.sql.DataFrame = [movieid: int, mean_rating: double, count: bigint]
```

Took 0 seconds (outdated)

FINISHED

```
avg_ratings_count.show( 5 )
```

```
+-------+------------------+-----+
|movieid|       mean_rating|count|
+-------+------------------+-----+
|     31|3.1134751773049647|  141|
|    231|3.1924242424242424|  660|
|    431|3.6910569105691056|  369|
|    631|              2.08|   75|
|    831|              3.85|   20|
+-------+-----------------+-----+
only showing top 5 rows
```

Took 61 seconds

FINISHED

```
// round off the mean_rating value to 2 decimal places
import scala.math.BigDecimal;

// define an udf (sql function) for rounding of
val roundto2places    = udf[Double, Double]( BigDecimal(_).setScale(2, BigDecimal.RoundingMode.HALF_UP).toDouble )

// Using the udf on the column to round off all values in the mean_rating column
avg_ratings_count = avg_ratings_count.withColumn( "mean_rating", roundto2places( avg_ratings_count("mean_rating") ) )
```

```
import scala.math.BigDecimal
roundto2places: org.apache.spark.sql.UserDefinedFunction = UserDefinedFunction(<function1>,DoubleType,List(DoubleType))
avg_ratings_count: org.apache.spark.sql.DataFrame = [movieid: int, mean_rating: double, count: bigint]
```

Took 2 seconds

FINISHED

```
// cachel the dataframe to prevent it from re-evaluting it everytime we are going to use it
avg_ratings_count.cache()
avg_ratings_count.show(5)
```

```
res88: org.apache.spark.sql.DataFrame = [movieid: int, mean_rating: double, count: bigint]
+-------+-----------+-----+
|movieid|mean_rating|count|
+-------+-----------+-----+
|     31|       3.11|  141|
|    231|       3.19|  660|
|    431|       3.69|  369|
|    631|       2.08|   75|
|    831|       3.85|   20|
+-------+-----------+-----+
only showing top 5 rows
```

Took 58 seconds

FINISHED

```
var avg_ratings_count_sorted = avg_ratings_count.sort( desc( "mean_rating" ) )
```

```
avg_ratings_count_sorted.cache()
avg_ratings_count_sorted.show(5)
```

```
avg_ratings_count_sorted: org.apache.spark.sql.DataFrame = [movieid: int, mean_rating: double, count: bigint]
res91: org.apache.spark.sql.DataFrame = [movieid: int, mean_rating: double, count: bigint]
+-------+-----------+-----+
|movieid|mean_rating|count|
+-------+-----------+-----+
|   3233|        5.0|    2|
|   3656|        5.0|    1|
|   3280|        5.0|    1|
|   3881|        5.0|    1|
|   3172|        5.0|    1|
+-------+-----------+-----+
only showing top 5 rows
```

Took 27 seconds

FINISHED

```
// Consider only those movies whih are atleast rated by 100 users. The avarage rating of this movie will be less biased
var avg_ratings_count_more_20 = avg_ratings_count_sorted.filter( avg_ratings_count("count") > 100 )
var popular_movies_ratings = avg_ratings_count_more_20.sort( desc( "mean_rating" ) , desc( "count") )
```

```
avg_ratings_count_more_20: org.apache.spark.sql.DataFrame = [movieid: int, mean_rating: double, count: bigint]
popular_movies_ratings: org.apache.spark.sql.DataFrame = [movieid: int, mean_rating: double, count: bigint]
```

Took 1 seconds (outdated)

FINISHED

```
// Read the movies data into a spark dataframe
var movies_df = sqlContext.read.format("com.databricks.spark.csv")
        .option("delimiter", "\t")
        .option("header", "true" )
        .option("inferSchema", "true")
        .load("file:///home/hadoop/lab/data/movies/movies.dat")
```

```
movies_df: org.apache.spark.sql.DataFrame = [movieid: int, name: string, tags: string]
```

Took 2 seconds

FINISHED

```
// cache and display the first 10 records
movies_df.cache()
movies_df.show( 10 )
```

```
res121: org.apache.spark.sql.DataFrame = [movieid: int, name: string, tags: string]
+-------+--------------------+--------------------+
|movieid|                name|                tags|
+-------+--------------------+--------------------+
|      1|    Toy Story (1995)|Animation|Childre...|
|      2|      Jumanji (1995)|Adventure|Childre...|
|      3|Grumpier Old Men ...|      Comedy|Romance|
|      4|Waiting to Exhale...|        Comedy|Drama|
|      5|Father of the Bri...|              Comedy|
|      6|         Heat (1995)|Action|Crime|Thri...|
|      7|      Sabrina (1995)|      Comedy|Romance|
|      8| Tom and Huck (1995)|Adventure|Children's|
|      9| Sudden Death (1995)|              Action|
|     10|    GoldenEye (1995)|Action|Adventure|...|
+-------+--------------------+--------------------+
only showing top 10 rows
```

Took 1 seconds

```
movies_df.printSchema()
```

FINISHED

```
root
 |-- movieid: integer (nullable = true)
 |-- name: string (nullable = true)
 |-- tags: string (nullable = true)
```

Took 0 seconds

```
var top_movies = popular_movies_ratings.limit(20).join( movies_df, "movieid" )
```

FINISHED

```
top_movies: org.apache.spark.sql.DataFrame = [movieid: int, mean_rating: double, count: bigint, name: string, tags: string]
```

Took 1 seconds

```
top_movies.show()
```

FINISHED

```
+-------+-----------+-----+--------------------+--------------------+
|movieid|mean_rating|count|                name|                tags|
+-------+-----------+-----+--------------------+--------------------+
|   2019|       4.56|  628|Seven Samurai (Th...|        Action|Drama|
|    318|       4.55| 2227|Shawshank Redempt...|               Drama|
|    858|       4.52| 2223|Godfather, The (1...|  Action|Crime|Drama|
|     50|       4.52| 1783|Usual Suspects, T...|      Crime|Thriller|
|    745|       4.52|  657|Close Shave, A (1...|Animation|Comedy|...|
|    527|       4.51| 2304|Schindler's List ...|           Drama|War|
|   1148|       4.51|  882|Wrong Trousers, T...|    Animation|Comedy|
|    922|       4.49|  470|Sunset Blvd. (a.k...|           Film-Noir|
|   1198|       4.48| 2514|Raiders of the Lo...|    Action|Adventure|
|    904|       4.48| 1050|  Rear Window (1954)|    Mystery|Thriller|
|   1178|       4.47|  230|Paths of Glory (1...|           Drama|War|
|    260|       4.45| 2991|Star Wars: Episod...|Action|Adventure|...|
|    750|       4.45| 1367|Dr. Strangelove o...|          Sci-Fi|War|
|   1212|       4.45|  480|Third Man, The (1...|    Mystery|Thriller|
|   1207|       4.43|  928|To Kill a Mocking...|               Drama|
|    720|       4.43|  438|Wallace & Gromit:...|           Animation|
|   3435|       4.42|  551|Double Indemnity ...|     Crime|Film-Noir|
|   2762|       4.41| 2459|Sixth Sense, The ...|            Thriller|
|    912|       4.41| 1669|   Casablanca (1942)|   Drama|Romance|War|
|    913|        4.4| 1043|Maltese Falcon, T...|   Film-Noir|Mystery|
+-------+-----------+-----+--------------------+--------------------+
```

Took 3 seconds

```
// Donot truncate the column values
top_movies.show( truncate = false )
```

```
+-------+-----------+-----+---------------------------------------------------------------------------+-------------------------------+
|movieid|mean_rating|count|name                                                                       |tags                           |
+-------+-----------+-----+---------------------------------------------------------------------------+-------------------------------+
|2019   |4.56       |628  |Seven Samurai (The Magnificent Seven) (Shichinin no samurai) (1954)        |Action|Drama                   |
|318    |4.55       |2227 |Shawshank Redemption, The (1994)                                           |Drama                          |
|858    |4.52       |2223 |Godfather, The (1972)                                                      |Action|Crime|Drama             |
|50     |4.52       |1783 |Usual Suspects, The (1995)                                                 |Crime|Thriller                 |
|745    |4.52       |657  |Close Shave, A (1995)                                                      |Animation|Comedy|Thriller      |
|527    |4.51       |2304 |Schindler's List (1993)                                                    |Drama|War                      |
|1148   |4.51       |882  |Wrong Trousers, The (1993)                                                 |Animation|Comedy               |
|922    |4.49       |470  |Sunset Blvd. (a.k.a. Sunset Boulevard) (1950)                              |Film-Noir                      |
|1198   |4.48       |2514 |Raiders of the Lost Ark (1981)                                             |Action|Adventure               |
|904    |4.48       |1050 |Rear Window (1954)                                                         |Mystery|Thriller               |
|1178   |4.47       |230  |Paths of Glory (1957)                                                      |Drama|War                      |
|260    |4.45       |2991 |Star Wars: Episode IV - A New Hope (1977)                                  |Action|Adventure|Fantasy|Sci-Fi|
|750    |4.45       |1367 |Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb (1963)|Sci-Fi|War                      |
|1212   |4.45       |480  |Third Man, The (1949)                                                      |Mystery|Thriller                |
|1207   |4.43       |928  |To Kill a Mockingbird (1962)                                               |Drama                           |
|720    |4.43       |438  |Wallace & Gromit: The Best of Aardman Animation (1996)                     |Animation                       |
|3435   |4.42       |551  |Double Indemnity (1944)                                                    |Crime|Film-Noir                 |
|2762   |4.41       |2459 |Sixth Sense, The (1999)                                                    |Thriller                        |
|912    |4.41       |1669 |Casablanca (1942)                                                          |Drama|Romance|War               |
|913    |4.4        |1043 |Maltese Falcon, The (1941)                                                 |Film-Noir|Mystery               |
+-------+-----------+-----+---------------------------------------------------------------------------+--------------------------------+
```

Took 1 seconds (outdated)

FINISHED

```
// Save the top 20 best rated movies in a file
top_movies.write.format("com.databricks.spark.csv")
    .option("header", "true")
    .save("file:///home/hadoop/lab/results/topmovies")
```

Took 1 seconds

FINISHED

```
%md

## Exercises
#### Find out 20 worst rated movies. But only consider those movies which are rated by at least 100 users.
#### Find out best 10 and worst 10 movies in the category of action and drams
```

Exercises

Find out 20 worst rated movies. But only consider those movies which are rated by at least 100 users.

Find out best 10 and worst 10 movies in the category of action and drams

Took 2 seconds

READY