Movie_recommendation

Cmd 1

```
1 print "sc type: ", type(sc)
2 print "Driver Program name: ", sc.appName
3 print "Spark version: ", sc.version
```

```
sc type:  <class '__main__.RemoteContext'>
Driver Program name:  Databricks Shell
Spark version:  2.1.1
```

Command took 0.16 seconds -- by meghana.rwgsql@gmail.com at 6/10/2017, 2:24:41 AM on Movies_Project

Cmd 2

```
1 moviesDF = spark.sql("select * from movies")
2 moviesDF.show()
```

▸ (1) Spark Jobs

```
+-------+--------------------+--------------------+
|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|
|     20|  Money Train (1995)|Action|Comedy|Cri...|
+-------+--------------------+--------------------+
only showing top 20 rows
```

Command took 3.26 seconds -- by meghana.rwgsql@gmail.com at 7/5/2017, 2:27:09 PM on Movies_Project

Cmd 3

```
1 spark.sql("SELECT count(*) FROM movies").show()
```

▸ (1) Spark Jobs

```
+--------+
|count(1)|
+--------+
|   27278|
+--------+
```

Command took 0.93 seconds -- by meghana.rwgsql@gmail.com at 7/5/2017, 2:27:29 PM on Movies_Project

Cmd 4

```
1 ratingsDF = spark.sql("select * from ratings")
2 ratingsDF.show()
```

▸ (1) Spark Jobs

```
+------+-------+------+----------+
|userId|movieId|rating| timestamp|
+------+-------+------+----------+
|     1|      2|   3.5|1112486027|
|     1|     29|   3.5|1112484676|
|     1|     32|   3.5|1112484819|
|     1|     47|   3.5|1112484727|
|     1|     50|   3.5|1112484580|
|     1|    112|   3.5|1094785740|
|     1|    151|   4.0|1094785734|
|     1|    223|   4.0|1112485573|
|     1|    253|   4.0|1112484940|
|     1|    260|   4.0|1112484826|
|     1|    293|   4.0|1112484703|
|     1|    296|   4.0|1112484767|
|     1|    318|   4.0|1112484798|
|     1|    337|   3.5|1094785709|
|     1|    367|   3.5|1112485980|
|     1|    541|   4.0|1112484603|
|     1|    589|   3.5|1112485557|
|     1|    593|   3.5|1112484661|
|     1|    653|   3.0|1094785691|
|     1|    919|   3.5|1094785621|
+------+-------+------+----------+
only showing top 20 rows
```

Command took 0.63 seconds -- by meghana.rwgsql@gmail.com at 7/5/2017, 2:27:48 PM on Movies_Project

Cmd 5

```
1 spark.sql("SELECT count(*) FROM ratings").show()
```

▸ (1) Spark Jobs

```
+--------+
|count(1)|
+--------+
|20000263|
+--------+
```

Command took 6.78 seconds -- by meghana.rwgsql@gmail.com at 7/5/2017, 3:27:10 PM on Movies_Project

Cmd 6

```
1 ratingsDF.printSchema()
```

```
root
 |-- userId: string (nullable = true)
 |-- movieId: string (nullable = true)
 |-- rating: string (nullable = true)
 |-- timestamp: string (nullable = true)
```

Command took 0.07 seconds -- by meghana.rwgsql@gmail.com at 6/27/2017, 10:00:32 PM on Project_Movies_Recommendation

Cmd 7

```
1 ratingsDF = ratingsDF.withColumn('rating',
  ratingsDF["rating"].cast("float"))
2 ratingsDF = ratingsDF.withColumn('userId', ratingsDF["userId"].cast("int"))
3 ratingsDF = ratingsDF.withColumn('movieId',
  ratingsDF["movieId"].cast("int"))
4 ratingsDF.show()
```

▸ (1) Spark Jobs

```
+------+-------+------+----------+
|userId|movieId|rating| timestamp|
+------+-------+------+----------+
|     1|      2|   3.5|1112486027|
|     1|     29|   3.5|1112484676|
|     1|     32|   3.5|1112484819|
|     1|     47|   3.5|1112484727|
|     1|     50|   3.5|1112484580|
|     1|    112|   3.5|1094785740|
|     1|    151|   4.0|1094785734|
|     1|    223|   4.0|1112485573|
|     1|    253|   4.0|1112484940|
|     1|    260|   4.0|1112484826|
|     1|    293|   4.0|1112484703|
|     1|    296|   4.0|1112484767|
```

```
|     1|    318|   4.0|1112484798|
|     1|    337|   3.5|1094785709|
|     1|    367|   3.5|1112485980|
|     1|    541|   4.0|1112484603|
|     1|    589|   3.5|1112485557|
|     1|    593|   3.5|1112484661|
|     1|    653|   3.0|1094785691|
|     1|    919|   3.5|1094785621|
+------+-------+------+----------+
only showing top 20 rows
```

Command took 0.48 seconds -- by meghana.rwgsql@gmail.com at 7/5/2017, 2:28:12 PM on Movies_Project

Cmd 8

```
1 from pyspark.sql.functions import from_unixtime
2 ratingsDF = ratingsDF.withColumn('timestamp',
  from_unixtime(ratingsDF.timestamp))
3
```

Command took 0.12 seconds -- by meghana.rwgsql@gmail.com at 7/5/2017, 2:28:20 PM on Movies_Project

Cmd 9

```
1 from pyspark.sql.types import TimestampType
2 ratingsDF = ratingsDF.withColumn('timestamp',
  ratingsDF.timestamp.cast(TimestampType()))
```

Command took 0.07 seconds -- by meghana.rwgsql@gmail.com at 7/5/2017, 2:28:24 PM on Movies_Project

Cmd 10

```
1 ratingsDF.printSchema()
```

```
root
 |-- userId: integer (nullable = true)
 |-- movieId: integer (nullable = true)
 |-- rating: float (nullable = true)
 |-- timestamp: timestamp (nullable = true)
```

Command took 0.07 seconds -- by meghana.rwgsql@gmail.com at 7/5/2017, 2:28:27 PM on Movies_Project

Cmd 11

```
1 moviesDF.describe().show()
```

▸ (1) Spark Jobs

```
+-------+-----------------+--------------------+------------------+
|summary|          movieId|               title|            genres|
+-------+-----------------+--------------------+------------------+
|  count|            27278|               27278|             27278|
```

```
|   mean|59855.48057042305|                null|              null|
| stddev|44429.31469707313|                null|              null|
|    min|                1|"""Great Performa...|(no genres listed)|
|    max|            99999|         貞子3D (2012)|           Western|
+-------+-----------------+--------------------+------------------+
```

Command took 2.12 seconds -- by meghana.rwgsql@gmail.com at 7/4/2017, 5:21:26 PM on Movies_Project

Cmd 12

```
1 #m_DF =
  sqlContext.read.format("csv").load("/FileStore/tables/4riqczku1497073456815
  /movies.csv")
2 #r_df =
  sqlContext.read.format("csv").load("/FileStore/tables/f9oap0qt1497073661552
  /ratings.csv")
```

▸ (2) Spark Jobs

Command took 0.93 seconds -- by meghana.rwgsql@gmail.com at 6/10/2017, 2:37:14 AM on Movies_Project

Cmd 13

```
1 ratingsDF.describe().show()
```

▸ (1) Spark Jobs

```
+-------+-----------------+-----------------+------------------+
|summary|           userId|          movieId|            rating|
+-------+-----------------+-----------------+------------------+
|  count|         20000263|         20000263|          20000263|
|   mean|69045.87258292554|9041.567330339605|3.5255285642993797|
| stddev|40038.62665316182|19789.47744541297|  1.05198891929425|
|    min|                1|                1|               0.5|
|    max|           138493|           131262|               5.0|
+-------+-----------------+-----------------+------------------+
```

Command took 12.91 seconds -- by meghana.rwgsql@gmail.com at 7/4/2017, 5:21:35 PM on Movies_Project

Cmd 14

```
1 ratingsDF.describe('rating', 'timestamp').show()
```

▸ (1) Spark Jobs

```
+-------+------------------+---------+
|summary|            rating|timestamp|
+-------+------------------+---------+
|  count|          20000263|        0|
|   mean|3.5255285642993797|     null|
| stddev|  1.05198891929425|     null|
|    min|               0.5|     null|
```

```
|    max|               5.0|     null|
+-------+------------------+---------+
```

Command took 1.28 minutes -- by meghana.rwgsql@gmail.com at 6/27/2017, 11:27:25 PM on Project_Movies_Recommendation

Cmd 15

```
1 #To check if Dataframe contains None/blank values
2 ratingsDF[ratingsDF.rating == None].count()
3 ratingsDF[ratingsDF.userId == None].count()
4 ratingsDF[ratingsDF.movieId == None].count()
```

▸ (3) Spark Jobs

Out[14]: 0

Command took 22.20 seconds -- by meghana.rwgsql@gmail.com at 7/5/2017, 3:08:52 PM on Movies_Project

Cmd 16

```
1 #Find all the movies that have average ratings greater than 4.0.
2 ratingsDF.groupBy("movieId")\
3 .avg("rating").filter("avg(rating) > 4.0")\
4 .show()
```

▸ (2) Spark Jobs

```
+-------+-----------------+
|movieId|      avg(rating)|
+-------+-----------------+
|  89056|            4.125|
|  78064|4.111111111111111|
|  81501|              4.5|
| 102119|              4.5|
|  97092|              4.5|
|    858|4.364732196832306|
|  48780|4.042195403318839|
| 118258|              4.5|
|   3226|4.666666666666667|
| 104317|              5.0|
|  97872|              4.5|
|   3000|4.096298619824341|
|   1303|4.040199611865816|
|  80906|4.005541346973572|
| 116183|              4.5|
|   3089|4.143596986817326|
| 104829|4.083333333333333|
|  56490|              4.1|
| 125599|              5.0|
|   1223|4.066765197275415|
+-------+-----------------+
```

```
only showing top 20 rows
```

Command took 10.31 seconds -- by meghana.rwgsql@gmail.com at 7/4/2017, 5:44:27 PM on Movies_Project

Cmd 17

```
1 #Find all the movies that have average ratings greater than 4.0 and have recieved more than 400 reviews
2
3 ratingsDF.groupBy("movieId")\
4 .agg({"rating":"avg", "*":"count"})\
5 .filter("avg(rating) > 4.0 AND count(1) > 400")\
6 .show()
```

▸ (3) Spark Jobs

```
+-------+-----------------+--------+
|movieId|      avg(rating)|count(1)|
+-------+-----------------+--------+
|    858|4.364732196832306|   41355|
|  48780|4.042195403318839|   11269|
|   3000|4.096298619824341|    9564|
|   1303|4.040199611865816|    3607|
|  80906|4.005541346973572|    1173|
|   3089|4.143596986817326|    3186|
|   1223|4.066765197275415|    7781|
|    898|4.171426401336777|    6583|
|   5995|4.053922967189729|   10515|
|    296|4.174231169217055|   67310|
|  68954|4.038266407599309|    9264|
|  86377| 4.08361391694725|     891|
|  58559|4.220129171151776|   20438|
|    593| 4.17705650958151|   63299|
|   1199|4.029590886293616|   13957|
|   1212|4.246001523229246|    6565|
|   7132|4.047279792746114|    1544|
|    950|   4.184187016081|    3358|
|   1198|4.219009123455364|   43295|
|   7587|4.065116279069767|     645|
+-------+-----------------+--------+
only showing top 20 rows
```

Command took 12.20 seconds -- by meghana.rwgsql@gmail.com at 7/4/2017, 5:25:39 PM on Movies_Project

Cmd 18

```
#Find all the movies that have less than 10 reviews

ratingsDF.groupBy("movieId")\
.agg({"*":"count"})\
.filter("count(1) <10")\
.show()
```

▸ (1) Spark Jobs

```
+-------+--------+
|movieId|count(1)|
+-------+--------+
| 104064|       5|
|  96469|       7|
| 104508|       2|
|  46952|       9|
| 102798|       8|
|  80451|       9|
|   7754|       4|
| 119432|       4|
|  89056|       4|
|  89844|       6|
|  92182|       2|
|  78064|       9|
|  80033|       3|
|  83250|       9|
|  81349|       2|
| 103747|       3|
|  71995|       5|
|  86927|       3|
|  53963|       5|
|  69042|       4|
+-------+--------+
only showing top 20 rows
```

Command took 10.35 seconds -- by meghana.rwgsql@gmail.com at 7/4/2017, 5:26:08 PM on Movies_Project

Cmd 19

```
# Colaborative filtering
# Splitting Data into Train and test
(training, test) = ratingsDF.randomSplit([0.8, 0.2])
```

Command took 0.07 seconds -- by meghana.rwgsql@gmail.com at 7/5/2017, 3:10:56 PM on Movies_Project

Cmd 20

```
training.show()
```

▸ (1) Spark Jobs

```
+------+-------+------+--------------------+
|userId|movieId|rating|           timestamp|
+------+-------+------+--------------------+
|     1|      2|   3.5|2005-04-02 23:53:...|
|     1|     29|   3.5|2005-04-02 23:31:...|
|     1|     47|   3.5|2005-04-02 23:32:...|
|     1|     50|   3.5|2005-04-02 23:29:...|
|     1|    112|   3.5|2004-09-10 03:09:...|
|     1|    151|   4.0|2004-09-10 03:08:...|
|     1|    253|   4.0|2005-04-02 23:35:...|
|     1|    260|   4.0|2005-04-02 23:33:...|
|     1|    293|   4.0|2005-04-02 23:31:...|
|     1|    296|   4.0|2005-04-02 23:32:...|
|     1|    318|   4.0|2005-04-02 23:33:...|
|     1|    367|   3.5|2005-04-02 23:53:...|
|     1|    541|   4.0|2005-04-02 23:30:...|
|     1|    593|   3.5|2005-04-02 23:31:...|
|     1|    653|   3.0|2004-09-10 03:08:...|
|     1|    919|   3.5|2004-09-10 03:07:...|
|     1|    924|   3.5|2004-09-10 03:06:...|
|     1|   1009|   3.5|2005-04-02 23:53:...|
|     1|   1036|   4.0|2005-04-02 23:44:...|
|     1|   1079|   4.0|2004-09-10 03:07:...|
+------+-------+------+--------------------+
only showing top 20 rows
```

Command took 15.41 seconds -- by meghana.rwgsql@gmail.com at 7/4/2017, 5:27:31 PM on Movies_Project

Cmd 21

```
1 #comparing Training and Test Dataset
2 tempDF = training.join(test, training.movieId == test.movieId)
3 display(tempDF)
```

▸ (1) Spark Jobs

userId	movieId	rating	timestamp
603	148	2	1996-07-25T15:28:36.000+0000
603	148	2	1996-07-25T15:28:36.000+0000
603	148	2	1996-07-25T15:28:36.000+0000
603	148	2	1996-07-25T15:28:36.000+0000
603	148	2	1996-07-25T15:28:36.000+0000
603	148	2	1996-07-25T15:28:36.000+0000
603	148	2	1996-07-25T15:28:36.000+0000

603	148	2	1996-07-25T15:28:36.000+0000

Showing the first 1000 rows.

Command took 50.24 seconds -- by meghana.rwgsql@gmail.com at 6/29/2017, 7:36:38 PM on Movies_project

Cmd 22

```
1 #Dataset in Training and not in Test
2 filtered_test = training.subtract(test)
3
4 display (filtered_test)
5 filtered_test.count()
```

▸ (2) Spark Jobs

userId	movieId	rating
13	509	4
14	31658	4
21	953	3
21	1282	3
21	4034	4
24	2539	2
24	4226	4
26	587	4
29	300	2

Showing the first 1000 rows.

Command took 1.90 minutes -- by meghana.rwgsql@gmail.com at 6/29/2017, 8:08:32 PM on Movies_project

Cmd 23

```
1 training.count()
```

▸ (1) Spark Jobs

Out[16]: 15998376

Command took 31.59 seconds -- by meghana.rwgsql@gmail.com at 7/5/2017, 3:11:06 PM on Movies_Project

Cmd 24

```
1 test.count()
```

▸ (1) Spark Jobs

Out[17]: 4001887

Command took 29.10 seconds -- by meghana.rwgsql@gmail.com at 7/5/2017, 3:25:39 PM on Movies_Project

Cmd 25

```
# Build the recommendation model using ALS on the training data
# Note we set cold start strategy to 'drop' to ensure we don't get NaN evaluation metrics
from pyspark.sql import SparkSession
from pyspark.ml.evaluation import RegressionEvaluator
from pyspark.ml.recommendation import ALS
from pyspark.sql import Row

als = ALS(rank=14, maxIter=20, regParam=0.01, userCol="userId", itemCol="movieId", ratingCol="rating", implicitPrefs=True)
#als = ALS(maxIter=5, regParam=0.01, userCol="userId", itemCol="movieId", ratingCol="rating", coldStartStrategy="drop")
model = als.fit(training)
```

▸ (3) Spark Jobs

Command took 3.25 minutes -- by meghana.rwgsql@gmail.com at 7/5/2017, 3:27:46 PM on Movies_Project

Cmd 26

```
#predictions = model.transform(test).dropna()
predictions = model.transform(test)
predictions = predictions.dropna()
predictions.registerTempTable("predictions_table")
```

Command took 0.12 seconds -- by meghana.rwgsql@gmail.com at 7/5/2017, 3:34:31 PM on Movies_Project

Cmd 27

```
#%sql select user_id, movie_id, rating, prediction from predictions
spark.sql("select userId, movieId, rating, prediction from predictions_table").show()

```

▸ (1) Spark Jobs

```
+------+-------+------+-----------+
|userId|movieId|rating| prediction|
+------+-------+------+-----------+
| 96393|    148|   3.0|0.020572461|
| 19067|    148|   2.0|  0.1497055|
| 96427|    148|   3.0| 0.09019219|
| 36821|    148|   4.0|  0.1357963|
| 83090|    148|   2.0| 0.28132528|
| 44882|    148|   4.0|  0.1479186|
| 80168|    148|   4.0|0.012591151|
| 36445|    148|   4.5|0.017339872|
```

```
|125969|    148|   3.0| 0.15357818|
|  3990|    148|   4.0|0.019826837|
| 46380|    148|   4.0| 0.07418445|
|108140|    148|   1.0| 0.10299908|
|  3673|    148|   2.0| 0.15612462|
|111523|    148|   2.0| 0.18260114|
| 67743|    148|   3.0|0.097946696|
|108929|    148|   1.0|0.071163654|
|135888|    148|   3.0|0.065951824|
| 68242|    148|   3.0|0.024129314|
|132268|    148|   2.0|0.056698002|
| 62028|    148|   3.0|0.050269227|
+------+-------+------+-----------+
only showing top 20 rows
```

Command took 35.71 seconds -- by meghana.rwgsql@gmail.com at 7/4/2017, 6:53:50 PM on Movies_Project

Cmd 28

```
1  # Evaluate the model by computing the RMSE on the test data
2 evaluator = RegressionEvaluator(metricName="rmse", labelCol="rating",
3                                 predictionCol="prediction")
4 rmse = evaluator.evaluate(predictions)
5 print("Root-mean-square error = " + str(rmse))
```

▸ (1) Spark Jobs

```
Root-mean-square error = 3.20042880226
```

Command took 42.53 seconds -- by meghana.rwgsql@gmail.com at 7/5/2017, 3:34:38 PM on Movies_Project

Cmd 29

```
1 displayHTML("<h4>The Root-mean-square error is %s</h4>" % str(rmse))
```

The Root-mean-square error is 3.20042880226

Command took 0.02 seconds -- by meghana.rwgsql@gmail.com at 7/5/2017, 4:44:26 PM on Movies_Project

Cmd 30

```
1
```

Cmd 31

```
1
```

Cmd 32

```
1
```

Cmd 33

```
DATA EXPLORATION AFTER MODELING
```

Cmd 34

```
#Data Exploration
#Explode movies table into title, year and categories
from pyspark.sql import Row

def create_row(line):
  atoms = line.split("::")
  movie = {"id": int(atoms[0])}
  year_begin = atoms[1].rfind("(")
  movie["name"] = atoms[1][0:year_begin].strip()
  movie["year"] = int(atoms[1][year_begin+1:-1])
  movie["categories"] = atoms[2].split("|")
  return Row(**movie)

movies_tbl = movies.map(create_row)
movies_tblDF = sqlContext.createDataFrame(movies_tbl)
movies_tblDF.registerTempTable("MoviesTable")
```

```
NameError: name 'movies' is not defined
```

Command took 0.06 seconds -- by meghana.rwgsql@gmail.com at 6/27/2017, 11:19:30 PM on Project_Movies_Recommendation

Cmd 35

```
#Select 10 random movies from the most rated, as those as likely to be commonly recognized movies. Create Databricks Widgets to allow a user to enter in ratings for those movies.

spark.sql("""
    select
      movieId, movies.name, count(*) as times_rated
    from
      ratings
    join
      movies on ratings.movieId = movies.movieId
    group by
      movie_id, movies.name, movies.year
    order by
      times_rated desc
    limit
      200
    """
).registerTempTable("most_rated_movies")
```

```
ERROR: An unexpected error occurred while tokenizing input
The following traceback may be corrupted or invalid
```

```
The error message is: ('EOF in multi-line string', (1, 4))

AnalysisException: u"cannot resolve '`ratings.movie_id`' given input columns: [g
enres, movieId, userId, movieId, timestamp, title, rating]; line 7 pos 16;\n'Glo
balLimit 200\n+- 'LocalLimit 200\n   +- 'Sort ['times_rated DESC NULLS LAST], tr
ue\n      +- 'Aggregate ['movie_id, 'movies.name, 'movies.year], ['movie_id, 'mo
vies.name, count(1) AS times_rated#718L]\n         +- 'Join Inner, ('ratings.mov
ie_id = 'movies.id)\n            :- SubqueryAlias ratings\n            :  +- Rel
ation[userId#19,movieId#20,rating#21,timestamp#22] csv\n            +- SubqueryA
lias movies\n               +- Relation[movieId#7,title#8,genres#9] csv\n"
```

Command took 0.26 seconds -- by meghana.rwgsql@gmail.com at 6/27/2017, 10:25:52 PM on Project_Movies_Recommendation

Cmd 36

```
if not "most_rated_movies" in vars():
  most_rated_movies = spark.table("most_rated_movies").rdd.takeSample(True, 10)
  for i in range(0, len(most_rated_movies)):
    dbutils.widgets.dropdown("movie_%i" % i, "5", ["1", "2", "3", "4", "5"], most_rated_movies[i].name)
```

Cmd 37

```
#Change the values on top to be your own personal ratings before proceeding.

from datetime import datetime
from pyspark.sql import Row
ratings = []
for i in range(0, len(most_rated_movies)):
  ratings.append(
    Row(user_id = 0,
        movie_id = most_rated_movies[i].movie_id,
        rating = int(dbutils.widgets.get("movie_%i" %i)),
        timestamp = datetime.now()))
myRatingsDF = spark.createDataFrame(ratings)
```

Cmd 38

```
from pyspark.sql.functions import explode

sqlContext.sql("select * from movies").select(
  "id", "name", "year", explode("categories").alias("category")
).registerTempTable("exploded_movies")
```

```
AnalysisException: u"cannot resolve '`id`' given input columns: [movieId, title,
 genres];;\n'Project ['id, 'name, 'year, explode('categories) AS category#179]\n
+- Project [movieId#7, title#8, genres#9]\n   +- SubqueryAlias movies\n      +-
 Relation[movieId#7,title#8,genres#9] csv\n"
```

Command took 0.36 seconds -- by meghana.rwgsql@gmail.com at 6/29/2017, 7:17:17 PM on Movies_project

Cmd 39