Naeem's Zeppline Notes

Check the Spark version first

FINISHED

- checking the Spark version first

Took 0 sec. Last updated by anonymous at February 05 2018, 8:11:14 PM.

Verifying Spark version

FINISHED

```
sc.version
```

```
res0: String = 2.2.0.2.6.3.0-235
```

Took 43 sec. Last updated by anonymous at February 05 2018, 8:15:04 PM. (outdated)

- Now lets import the movie ratings and movies data using shell command wget.

FINISHED

Took 0 sec. Last updated by anonymous at February 05 2018, 8:23:08 PM.

Downloading data to local machine

FINISHED

```
%sh
wget https://s3.amazonaws.com/testbucket786786/u.data -O /tmp/u.data
wget https://s3.amazonaws.com/testbucket786786/u.item -O /tmp/u.item
```

```
 1650K .......... .......... .......... .......... .......... 87% 1.59M 0s
 1700K .......... .......... .......... .......... .......... 90% 12.4M 0s
 1750K .......... .......... .......... .......... .......... 93% 9.64M 0s
 1800K .......... .......... .......... .......... .......... 95% 1.49M 0s
 1850K .......... .......... .......... .......... .......... 98% 15.7M 0s
 1900K .......... .......... .......... ..                   100% 4.07M=1.5s
2018-02-06 02:37:02 (1.26 MB/s) - “/tmp/u.data” saved [1979226/1979226]
--2018-02-06 02:37:02--  https://s3.amazonaws.com/testbucket786786/u.item
Resolving s3.amazonaws.com... 52.216.80.211
Connecting to s3.amazonaws.com|52.216.80.211|:443... connected.
HTTP request sent, awaiting response... 200 OK
```

```
Length: 236344 (231K) [binary/octet-stream]
Saving to: “/tmp/u.item”
     0K ......... ......... ......... ......... ......... 21%  418K 0s
    50K ......... ......... ......... ......... ......... 43%  900K 0s
   100K ......... ......... ......... ......... ......... 64% 1.33M 0s
   150K ......... ......... ......... ......... ......... 86% 1.44M 0s
   200K                                                    100% 2.18M 0.2s
```

Took 12 sec. Last updated by anonymous at February 05 2018, 8:37:07 PM. (outdated)

- Now lets upload the data to HDFS

FINISHED

Took 0 sec. Last updated by anonymous at February 05 2018, 8:34:02 PM.

Uploading data from local to HDFS

FINISHED

```
%sh
hadoop fs -rm -r -f /tmp/ml-100k
hadoop fs -mkdir /tmp/ml-100k
hadoop fs -put /tmp/u.data /tmp/ml-100k/
hadoop fs -put /tmp/u.item /tmp/ml-100k/
```

```
18/02/06 02:40:04 INFO fs.TrashPolicyDefault: Moved: 'hdfs://sandbox-hdp.hortonworks.com:8020/tmp/ml-100k' to trash at: hdfs://sandbox-hdp.hortonworks.com:8020/user/zeppelin/.Trash/Current/tmp/ml-100k
```

Took 14 sec. Last updated by anonymous at February 05 2018, 8:40:15 PM. (outdated)

Using Scale code to read the lines by splitting by delimiter tab and pipe for both files respectively

FINISHED

```
final case class MovieRating(movieID: Int, rating: Int)
val ratinglines = sc.textFile("hdfs:///tmp/ml-100k/u.data").map(x => {val fields = x.split("\t"); MovieRating(fields(1).toInt, fields(2)
final case class Movies(movieID: Int, movieName: String)
val movielines = sc.textFile("hdfs:///tmp/ml-100k/u.item").map(x => {val fields = x.split('|'); Movies(fields(0).toInt, fields(1))})
```

```
warning: there was one unchecked warning; re-run with -unchecked for details
defined class MovieRating
ratinglines: org.apache.spark.rdd.RDD[MovieRating] = MapPartitionsRDD[82] at map at <console>:43
warning: there was one unchecked warning; re-run with -unchecked for details
defined class Movies
movielines: org.apache.spark.rdd.RDD[Movies] = MapPartitionsRDD[85] at map at <console>:43
```

Took 1 sec. Last updated by anonymous at February 05 2018, 9:36:51 PM. (outdated)

Converting the files into a Data Frame

FINISHED

```
import sqlContext.implicits._
val movieRatingsDF = ratinglines.toDF()
movieRatingsDF.printSchema()
val moviesDF = movielines.toDF()
moviesDF.printSchema()
```

```
import sqlContext.implicits._
movieRatingsDF: org.apache.spark.sql.DataFrame = [movieID: int, rating: int]
root
 |-- movieID: integer (nullable = false)
 |-- rating: integer (nullable = false)
moviesDF: org.apache.spark.sql.DataFrame = [movieID: int, movieName: string]
root
 |-- movieID: integer (nullable = false)
 |-- movieName: string (nullable = true)
```

Took 2 sec. Last updated by anonymous at February 05 2018, 9:36:57 PM. (outdated)

Lets group by the data by movieID and count and order by descending order of the count and also see movies data

FINISHED

```
val myTopMovieIDs = movieRatingsDF.groupBy("movieID").count().orderBy(desc("count")).cache()
myTopMovieIDs.show()
val myMovieIDs = moviesDF.cache()
myMovieIDs.show()
```

```
|      4|   Get Shorty (1995)|
|      5|      Copycat (1995)|
|      6|Shanghai Triad (Y...|
|      7|Twelve Monkeys (1...|
|      8|         Babe (1995)|
|      9|Dead Man Walking ...|
|     10|  Richard III (1995)|
|     11|Seven (Se7en) (1995)|
|     12|Usual Suspects, T...|
|     13|Mighty Aphrodite ...|
|     14|  Postino, Il (1994)|
|     15|Mr. Holland's Opu...|
|     16|French Twist (Gaz...|
|     17|From Dusk Till Da...|
|     18|White Balloon, Th...|
|     19|Antonia's Line (1...|
```

```
|     20|Angels and Insect...|
+-------+--------------------+
```

Took 6 sec. Last updated by anonymous at February 05 2018, 9:37:07 PM. (outdated)

Now register the Data Frames to a table

FINISHED

```
movieRatingsDF.registerTempTable("ratings")
moviesDF.registerTempTable("movies")
```

```
warning: there was one deprecation warning; re-run with -deprecation for details
warning: there was one deprecation warning; re-run with -deprecation for details
```

Took 1 sec. Last updated by anonymous at February 05 2018, 9:39:21 PM. (outdated)

Verify the data in ratings table using select statement

FINISHED

```
%sql
select * from ratings limit 10
```

movieID	rating
172	5
133	1
242	3
302	3
377	1
51	2
346	1
474	4

Took 0 sec. Last updated by anonymous at February 05 2018, 9:40:00 PM.

Viewing the ratings group by their count

FINISHED

```
%sql
select rating, count(*) as numCount from ratings group by rating
```

rating	numCount
1	6111
3	27145
5	21203
4	34174
2	11370

Took 2 sec. Last updated by anonymous at February 05 2018, 9:43:21 PM.

Viewing the movieID group by their ratingcount

FINISHED

```
%sql
select movieID, count(*) as ratingCount from ratings group by movieID order by ratingCount desc
```

movieID	ratingCount
302	297
328	295

96	295
15	293
25	293
118	293
183	291

Output is truncated to 1000 rows. Learn more about **zeppelin.spark.maxResult**

Took 2 sec. Last updated by anonymous at February 05 2018, 9:43:40 PM.

View the movie table

FINISHED

```
%sql
select * from movies limit 10
```

movieID	movieName
2	GoldenEye (1995)
3	Four Rooms (1995)
4	Get Shorty (1995)
5	Copycat (1995)
6	Shanghai Triad (Yao a yao yao dao waipo qiao) (1995)
7	Twelve Monkeys (1995)
8	Babe (1995)
9	Dead Man Walking (1995)

Took 0 sec. Last updated by anonymous at February 05 2018, 9:40:28 PM. (outdated)

Lets join the tables to view meaningful info

FINISHED

```
%sql
select m.movieName as title, count(*) as ratingCount
from movies m join ratings r on m.movieID=r.movieID
group by title
order by ratingCount DESC
```

title
Boogie Nights (1997)
Devil's Advocate, The (1997)
Lone Star (1996)
Long Kiss Goodnight, The (1996)
Ulee's Gold (1997)
Mighty Aphrodite (1995)
Postino, Il (1994)

Output is truncated to 1000 rows. Learn more about **zeppelin.spark.maxResult**

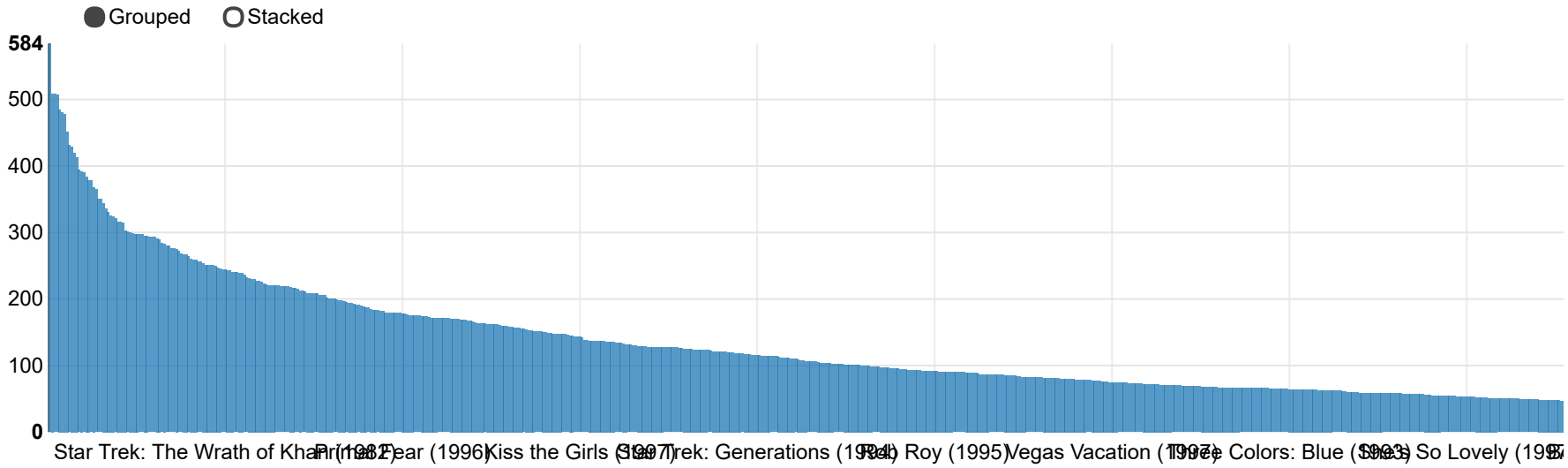
Took 5 sec. Last updated by anonymous at February 05 2018, 9:46:25 PM.

Lets see a graphical chart

FINISHED

```
%sql
select m.movieName as title, count(*) as ratingCount
from movies m join ratings r on m.movieID=r.movieID
group by title
order by ratingCount DESC
```

settings



Output is truncated to 1000 rows. Learn more about **zeppelin.spark.maxResult**

Took 2 sec. Last updated by anonymous at February 05 2018, 9:48:59 PM. (outdated)

```
%sql
```

READY