```
!apt-get install openjdk-8-jdk-headless -qq > /dev/null
!wget -q https://apache.mirror.colo-serv.net/spark/spark-2.4
!tar xf spark-2.4.7-bin-hadoop2.7.tgz
!pip install -q findspark

import os
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64
os.environ["SPARK_HOME"] = "/content/spark-2.4.7-bin-hadoop2

import findspark
findspark.init("spark-2.4.7-bin-hadoop2.7")# SPARK_HOME

import pyspark
from pyspark.sql import *
from pyspark.sql.functions import *
from pyspark import SparkContext, SparkConf

sc = SparkContext.getOrCreate()
spark = SparkSession.builder.getOrCreate()


!wget http://files.grouplens.org/datasets/movielens/ml-lates
!unzip ml-latest-small.zip
```

```
--2021-04-09 05:53:39--  http://files.grouplens.org/dat
Resolving files.grouplens.org (files.grouplens.org)...
Connecting to files.grouplens.org (files.grouplens.org)
HTTP request sent, awaiting response... 200 OK
Length: 978202 (955K) [application/zip]
Saving to: ‘ml-latest-small.zip’

ml-latest-small.zip 100%[===================>] 955.28K

2021-04-09 05:53:39 (6.24 MB/s) - ‘ml-latest-small.zip’

Archive:  ml-latest-small.zip
   creating: ml-latest-small/
```

```
      inflating: ml-latest-small/links.csv
      inflating: ml-latest-small/tags.csv
      inflating: ml-latest-small/ratings.csv
      inflating: ml-latest-small/README.txt
      inflating: ml-latest-small/movies.csv
```

```
ratings = spark.read.csv("ml-latest-small/ratings.csv", head
movies = spark.read.csv("ml-latest-small/movies.csv", header
```

```
ratings.printSchema()
movies.printSchema()
```

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

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

```
ratings.take(3)
```

```
    [Row(userId='1', movieId='1', rating='4.0', timestamp='
     Row(userId='1', movieId='3', rating='4.0', timestamp='
     Row(userId='1', movieId='6', rating='4.0', timestamp='
```

```
ratings.show()
```

```
    +------+-------+------+---------+
    |userId|movieId|rating|timestamp|
    +------+-------+------+---------+
    |     1|      1|   4.0|964982703|
    |     1|      3|   4.0|964981247|
    |     1|      6|   4.0|964982224|
    |     1|     47|   5.0|964983815|
    |     1|     50|   5.0|964982931|
    |     1|     70|   3.0|964982400|
    |     1|    101|   5.0|964980868|
    |     1|    110|   4.0|964982176|
    |     1|    151|   5.0|964984041|
    |     1|    157|   5.0|964984100|
    |     1|    163|   5.0|964983650|
    |     1|    216|   5.0|964981208|
    |     1|    223|   3.0|964980985|
    |     1|    231|   5.0|964981179|
    |     1|    235|   4.0|964980908|
    |     1|    260|   5.0|964981680|
    |     1|    296|   3.0|964982967|
    |     1|    316|   3.0|964982310|
    |     1|    333|   5.0|964981179|
    |     1|    349|   4.0|964982563|
    +------+-------+------+---------+
    only showing top 20 rows
```

```
movies.show()
```

```
+-------+--------------------+--------------------+
|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
```

```
print("In total there are {0} movies".format(movies.count()))
```

```
In total there are 9742 movies
```

```
# TODO
# How many ratings are there for each movie id?
# Sort descending by the count of ratings.
# Do not use SQL for this cell.
```

```
movie_counts = ratings.groupBy("movieId")\
                      .agg(count("rating")\
                      .alias("cnt"))\
                      .sort(desc("cnt"))

movie_counts.show()
```

```
+-------+---+
|movieId|cnt|
+-------+---+
|    356|329|
|    318|317|
|    296|307|
|    593|279|
|   2571|278|
|    260|251|
|    480|238|
|    110|237|
|    589|224|
|    527|220|
|   2959|218|
|      1|215|
|   1196|211|
|   2858|204|
|     50|204|
|     47|203|
|    780|202|
|    150|201|
|   1198|200|
|   4993|198|
+-------+---+
only showing top 20 rows
```

```
# TODO
# Register the ratings and movies dataframes as tables
# so that we can execute sql queries on them.
ratings.registerTempTable('ratings')
movies.registerTempTable('movies')
```

```
# TODO
# Use SQL
# How many ratings are there for each movie id?
# Sort descending by the count of ratings.
query ="""
select r.movieId, count(r.rating) as cnt
from ratings r
group by r.movieId
order by cnt desc
"""
result = spark.sql(query)
result = spark.sql(query)
result.show()
```

```
    +-------+---+
    |movieId|cnt|
    +-------+---+
    |    356|329|
    |    318|317|
    |    296|307|
    |    593|279|
    |   2571|278|
    |    260|251|
    |    480|238|
    |    110|237|
    |    589|224|
    |    527|220|
    |   2959|218|
    |      1|215|
    |   1196|211|
    |   2858|204|
    |     50|204|
    |     47|203|
    |    780|202|
    |    150|201|
    |   1198|200|
    |   4993|198|
    +-------+---+
    only showing top 20 rows
```

```
# TODO
# Use SQL
# Find the average rating for each movie that has more than
# Sort descending by average rating.

query ="""
select movieId, title, avg(rating) as avgrating
from ratings natural join movies
```

```
group by movieId, title
having count(rating) >50
order by avgrating desc
"""
result = spark.sql(query)
result.show()
```

```
    +-------+--------------------+------------------+
    |movieId|               title|         avgrating|
    +-------+--------------------+------------------+
    |    318|Shawshank Redempt...| 4.429022082018927|
    |    858|Godfather, The (1...|         4.2890625|
    |   2959|   Fight Club (1999)| 4.272935779816514|
    |   1276|Cool Hand Luke (1...| 4.271929824561403|
    |    750|Dr. Strangelove o...| 4.268041237113402|
    |    904|  Rear Window (1954)| 4.261904761904762|
    |   1221|Godfather: Part I...|  4.25968992248062|
    |  48516|Departed, The (2006)| 4.252336448598131|
    |   1213|   Goodfellas (1990)|              4.25|
    |    912|   Casablanca (1942)|              4.24|
    |  58559|Dark Knight, The ...| 4.238255033557047|
    |     50|Usual Suspects, T...| 4.237745098039215|
    |   1197|Princess Bride, T...| 4.232394366197183|
    |    260|Star Wars: Episod...| 4.231075697211155|
    |    527|Schindler's List ...|             4.225|
    |   1208|Apocalypse Now (1...| 4.219626168224299|
    |   2329|American History ...| 4.217054263565892|
    |   1196|Star Wars: Episod...|4.2156398104265405|
    |   1252|    Chinatown (1974)| 4.211864406779661|
    |   1198|Raiders of the Lo...|            4.2075|
    +-------+--------------------+------------------+
    only showing top 20 rows
```

```
movies.show()
```

```
    +-------+--------------------+--------------------+
    |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
```

```
# TODO
# Here we want to extract the genres from the movies table.
# This can be done easily using the underlying RDD of the mo

genres_rdd = movies.rdd.map(lambda row: (row.movieId, row.ge

genres_rdd.take(20)
```

```
[('1', 'Adventure|Animation|Children|Comedy|Fantasy'),
 ('2', 'Adventure|Children|Fantasy'),
 ('3', 'Comedy|Romance'),
 ('4', 'Comedy|Drama|Romance'),
 ('5', 'Comedy'),
 ('6', 'Action|Crime|Thriller'),
 ('7', 'Comedy|Romance'),
 ('8', 'Adventure|Children'),
 ('9', 'Action'),
 ('10', 'Action|Adventure|Thriller'),
 ('11', 'Comedy|Drama|Romance'),
 ('12', 'Comedy|Horror'),
 ('13', 'Adventure|Animation|Children'),
 ('14', 'Drama'),
 ('15', 'Action|Adventure|Romance'),
 ('16', 'Crime|Drama'),
 ('17', 'Drama|Romance'),
 ('18', 'Comedy'),
 ('19', 'Comedy'),
 ('20', 'Action|Comedy|Crime|Drama|Thriller')]
```

```
# TODO
# Now we want to create a flattened out RDD of (movieId, gen
# Use flatMapValues on the previous RDD.
# See: https://spark.apache.org/docs/1.1.1/api/python/pyspar
# Call the result RDD: moviegenres_flat_rdd
def f(genres_rdd) : return genres_rdd.split('|')
moviegenres_flat_rdd=genres_rdd.flatMapValues(f).collect()
genres_rdd.flatMapValues(f).take(20)
```

```
    [('1', 'Adventure'),
     ('1', 'Animation'),
     ('1', 'Children'),
     ('1', 'Comedy'),
     ('1', 'Fantasy'),
     ('2', 'Adventure'),
     ('2', 'Children'),
     ('2', 'Fantasy'),
     ('3', 'Comedy'),
     ('3', 'Romance'),
     ('4', 'Comedy'),
     ('4', 'Drama'),
     ('4', 'Romance'),
     ('5', 'Comedy'),
     ('6', 'Action'),
     ('6', 'Crime'),
     ('6', 'Thriller'),
     ('7', 'Comedy'),
     ('7', 'Romance'),
     ('8', 'Adventure')]


# TODO
# Now convert the last RDD into a dataframe for further anal
# Just uncomment the following lines.

moviegenres = spark.createDataFrame(moviegenres_flat_rdd).to
moviegenres.show()

# Register dataframe as a table.
moviegenres.registerTempTable('moviegenres')
```

```
     +-------+---------+
     |movieId|    genre|
     +-------+---------+
     |      1|Adventure|
     |      1|Animation|
     |      1| Children|
     |      1|   Comedy|
     |      1|  Fantasy|
     |      2|Adventure|
     |      2| Children|
     |      2|  Fantasy|
     |      3|   Comedy|
     |      3|  Romance|
     |      4|   Comedy|
     |      4|    Drama|
     |      4|  Romance|
     |      5|   Comedy|
     |      6|   Action|
     |      6|    Crime|
     |      6| Thriller|
     |      7|   Comedy|
     |      7|  Romance|
     |      8|Adventure|
     +-------+---------+
     only showing top 20 rows
```

```
# TODO
# Use SQL
# Find the average rating for each genre that has at least 5
# Order descending by average rating.
query ="""
select  genre, avg(rating) as avgrating
from ratings natural join moviegenres
group by genre
having count(rating) >50
order by avgrating desc
"""
result = spark.sql(query)
result.show()
```

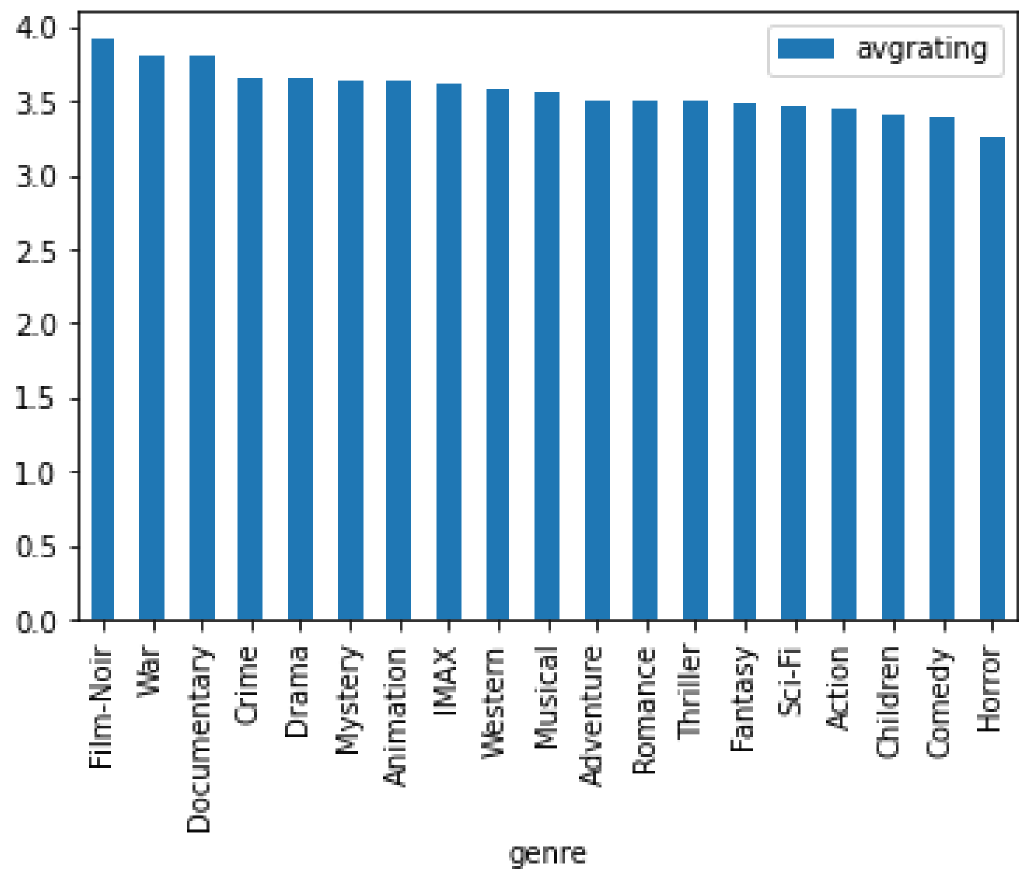
```
    +-----------+------------------+
    |      genre|         avgrating|
    +-----------+------------------+
    |  Film-Noir| 3.920114942528736|
    |        War|   3.8082938876312|
    |Documentary| 3.797785069729286|
    |      Crime| 3.658293867274144|
    |      Drama|3.6561844113718758|
    |    Mystery| 3.632460255407871|
    |  Animation|3.6299370349170004|
    |       IMAX| 3.618335343787696|
    |    Western| 3.583937823834197|
    |    Musical|3.5636781053649105|
    |  Adventure|3.5086089151939075|
    |    Romance|3.5065107040388437|
    |   Thriller|3.4937055799183425|
    |    Fantasy|3.4910005070136894|
    |     Sci-Fi| 3.455721162210752|
    |     Action| 3.447984331646809|
    |   Children| 3.412956125108601|
    |     Comedy|3.3847207640898267|
    |     Horror| 3.258195034974626|
    +-----------+------------------+
```

```
# Uncomment following lines.
result_pd = result.toPandas()
result_pd.head()
```

	genre	avgrating
0	Film-Noir	3.920115
1	War	3.808294
2	Documentary	3.797785
3	Crime	3.658294
4	Drama	3.656184

```
# Uncomment following line.
result_pd.plot(kind="bar", x="genre", y="avgrating")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f15d162eb9
```


avgrating
Film-Noir
War
Documentary
Crime
Drama
Mystery
Animation
IMAX
Western
Musical
Adventure
Romance
Thriller
Fantasy
Sci-Fi
Action
Children
Comedy
Horror
genre