BIG DATA – UE18CS322

FINAL PROJECT REPORT

TOPIC :- MOVIE DATA ANALYSIS

PROJECT BY :-

SINDHURA S – PES2201800374

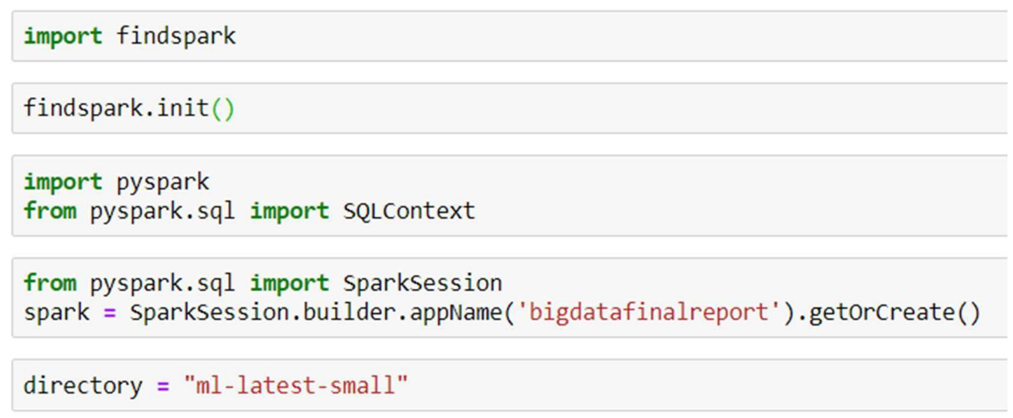
ABSTARCT

In real world prediction models and mechanisms can be used to predict the success of a movie. The proposed work aims to develop a system based upon data mining techniques that may help in predicting the success of a movie in advance thereby reducing certain level of uncertainty. An attempt is made to predict the past as well as the future of movie for the purpose of business certainty or simply a theoretical condition in which decision making [the success of the movie] is without risk, because the decision maker [movie makers and stake holders] has all the information about the exact outcome of the decision, before he or she makes the decision [release of the movie]. With over two million spectators a day and films exported to over 100 countries, the impact of Bollywood film industry is formidable We gather a series of interesting facts and relationships using a variety of data mining techniques.The system here is built using pyspark.

DATASET

The dataset contains details and ratings of the movie and it has FOUR CSV files and here we use only TWO CSV files, they are movies.csv and ratings.csv.

Initializing Findspark and start a spark session using the SparkSession builder .



```
import findspark
```

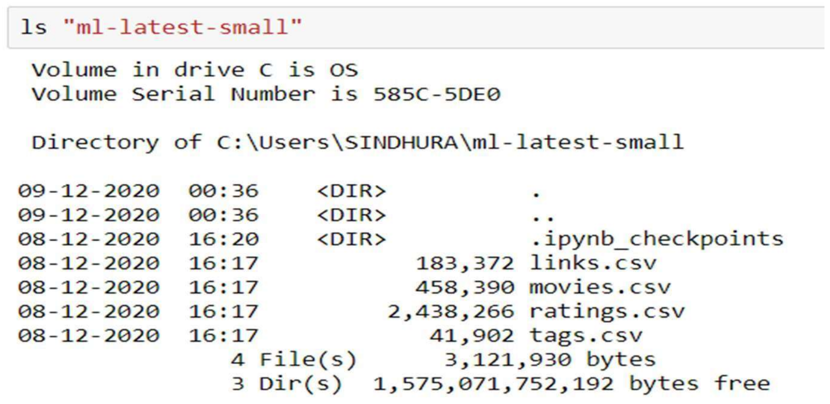
```
findspark.init()
```

```
import pyspark
from pyspark.sql import SQLContext
```

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName('bigdatafinalreport').getOrCreate()
```

```
directory = "ml-latest-small"
```

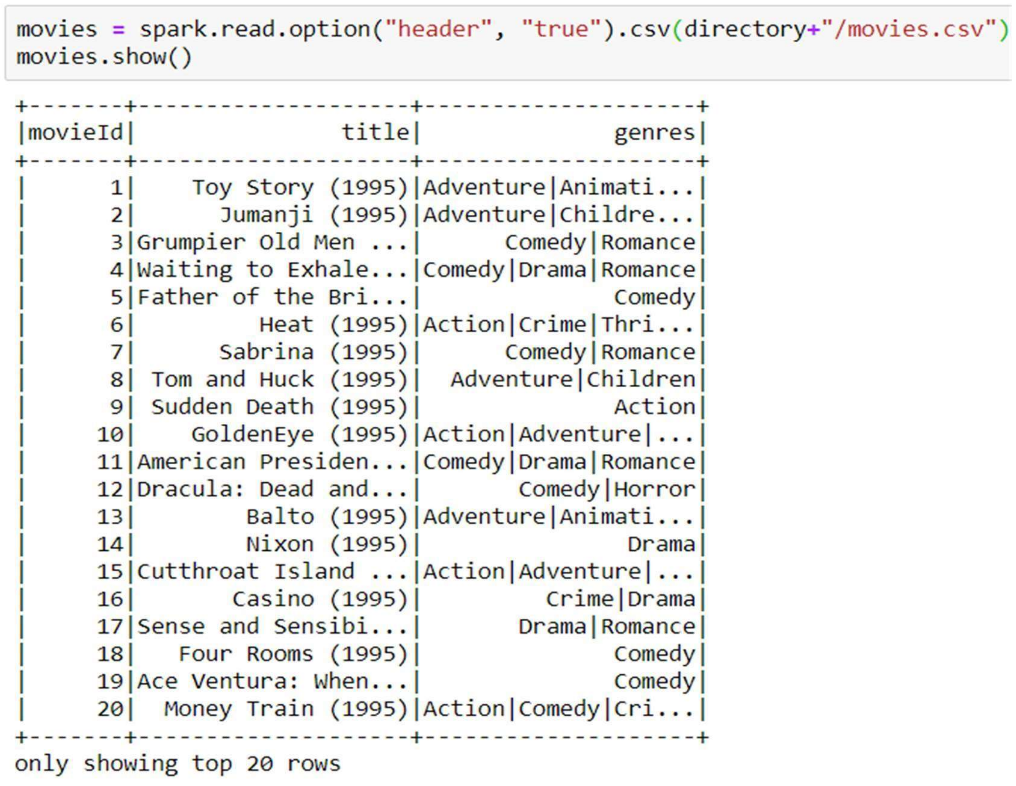
ls = “ml-latest-small” shows the CSV files



```
ls "ml-latest-small"
```

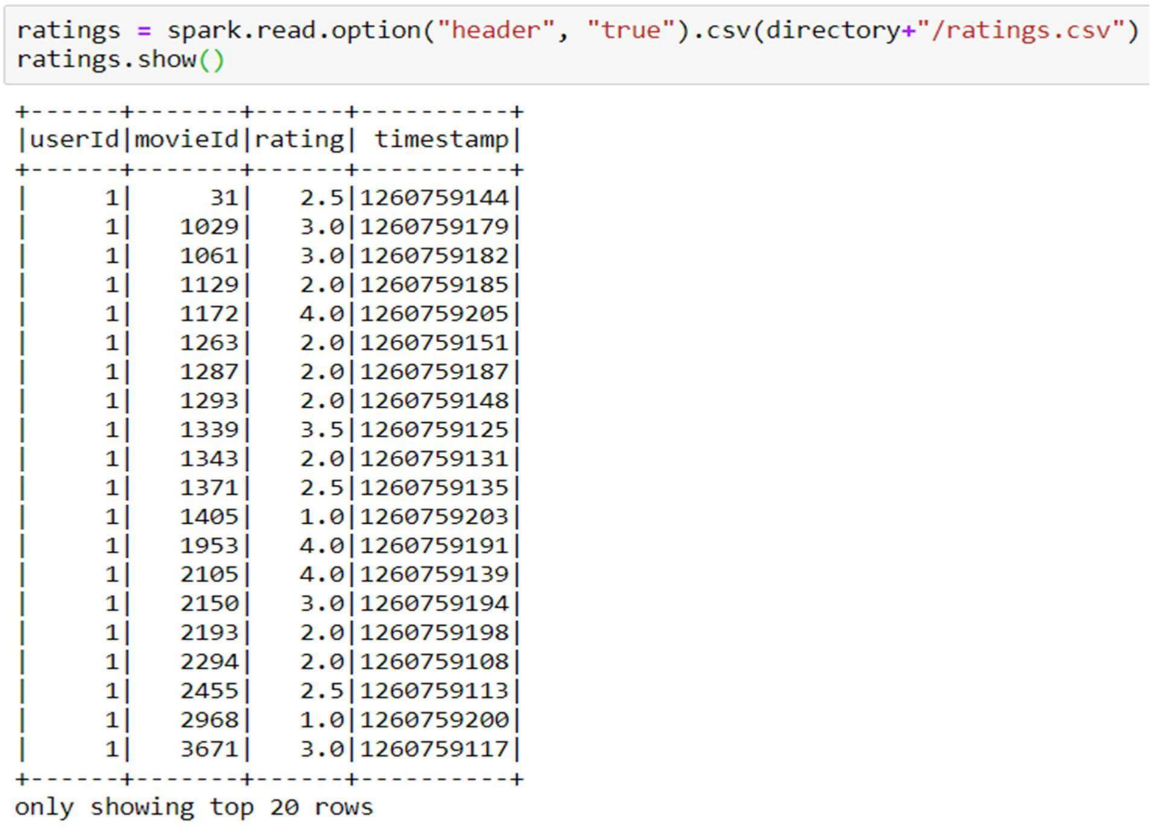
```
 Volume in drive C is OS
 Volume Serial Number is 585C-5DE0

 Directory of C:\Users\SINDHURA\ml-latest-small

09-12-2020  00:36    <DIR>          .
09-12-2020  00:36    <DIR>          ..
08-12-2020  16:20    <DIR>          .ipynb_checkpoints
08-12-2020  16:17           183,372 links.csv
08-12-2020  16:17           458,390 movies.csv
08-12-2020  16:17         2,438,266 ratings.csv
08-12-2020  16:17            41,902 tags.csv
               4 File(s)      3,121,930 bytes
               3 Dir(s)  1,575,071,752,192 bytes free
```

READING THE DATASET



```
movies = spark.read.option("header", "true").csv(directory+"/movies.csv")
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
```

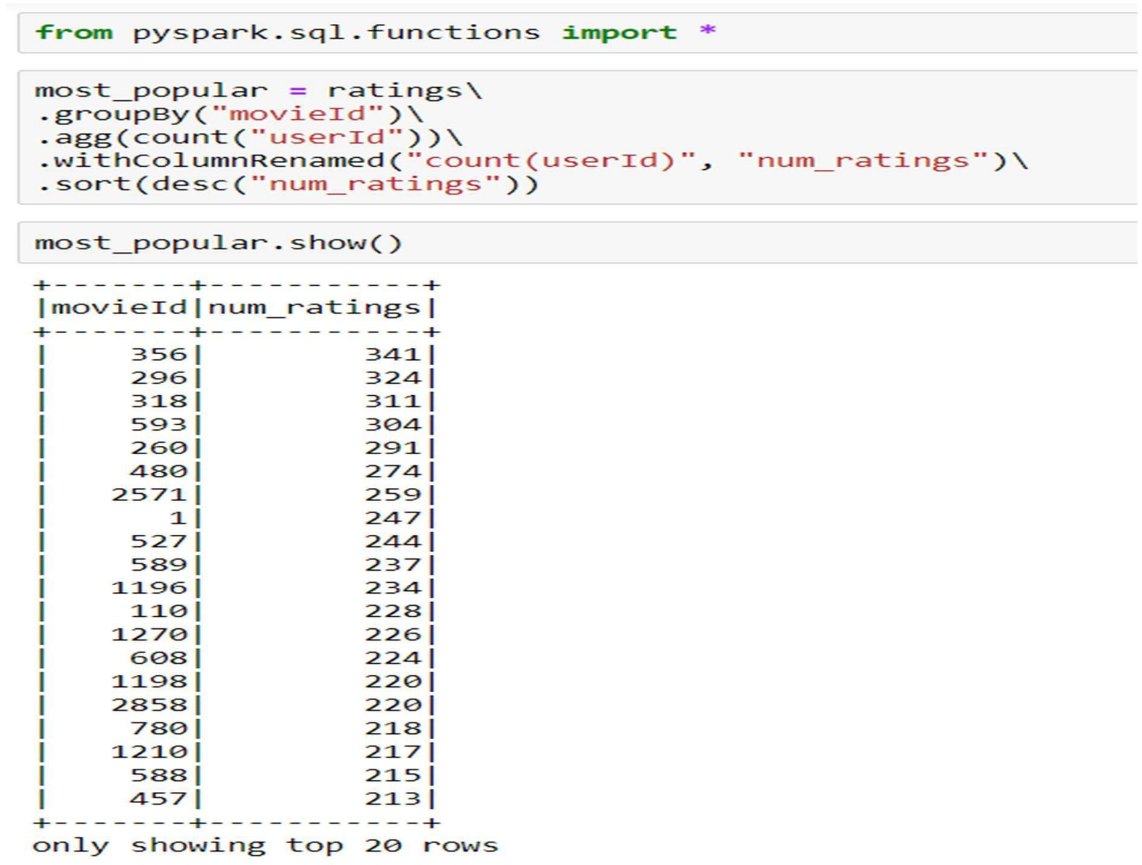


```
ratings = spark.read.option("header", "true").csv(directory+"/ratings.csv")
ratings.show()
```

```
+------+-------+------+----------+
|userId|movieId|rating| timestamp|
+------+-------+------+----------+
|     1|     31|   2.5|1260759144|
|     1|   1029|   3.0|1260759179|
|     1|   1061|   3.0|1260759182|
|     1|   1129|   2.0|1260759185|
|     1|   1172|   4.0|1260759205|
|     1|   1263|   2.0|1260759151|
|     1|   1287|   2.0|1260759187|
|     1|   1293|   2.0|1260759148|
|     1|   1339|   3.5|1260759125|
|     1|   1343|   2.0|1260759131|
|     1|   1371|   2.5|1260759135|
|     1|   1405|   1.0|1260759203|
|     1|   1953|   4.0|1260759191|
|     1|   2105|   4.0|1260759139|
|     1|   2150|   3.0|1260759194|
|     1|   2193|   2.0|1260759198|
|     1|   2294|   2.0|1260759108|
|     1|   2455|   2.5|1260759113|
|     1|   2968|   1.0|1260759200|
|     1|   3671|   3.0|1260759117|
+------+-------+------+----------+
only showing top 20 rows
```

EXPLORE THE DATASET

Finding the most popular movies by counting the number of ratings a movie has received. We make use of Pyspark SQL functions to get the most popular movies.

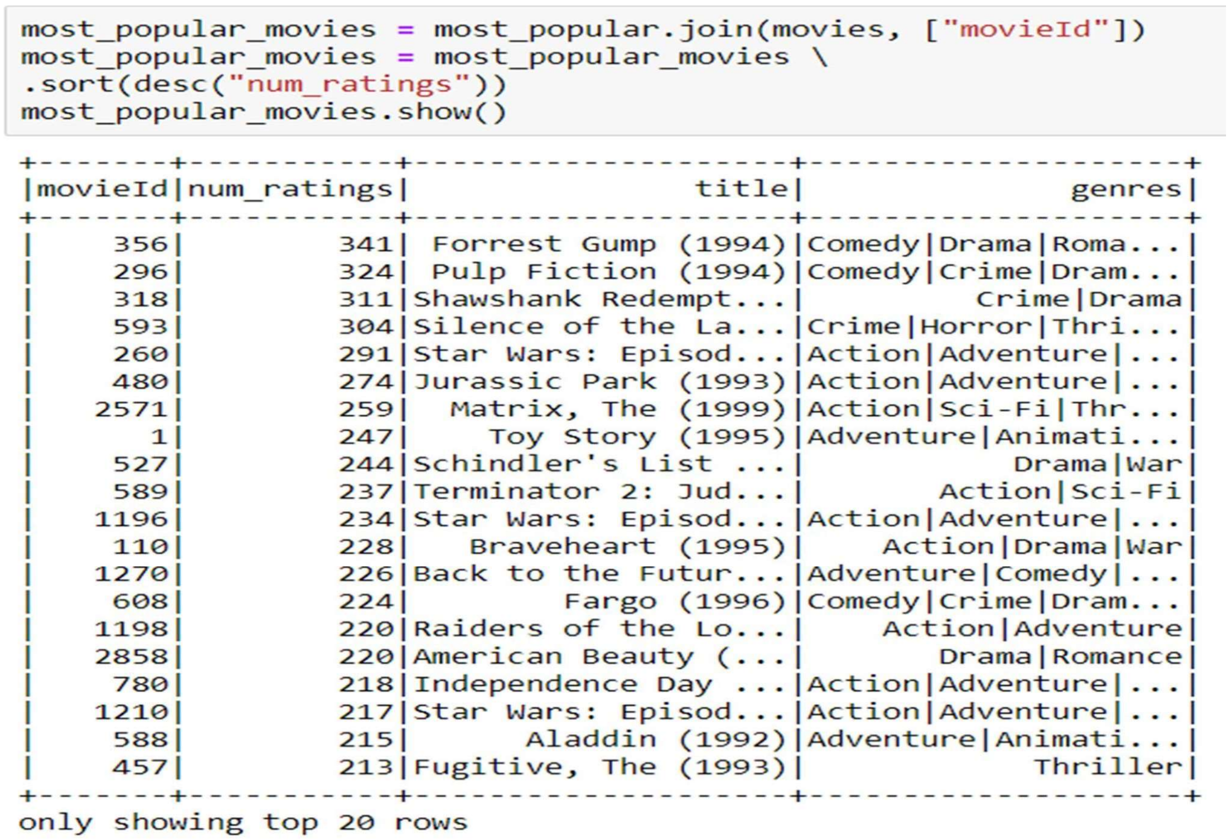


```
from pyspark.sql.functions import *
```

```
most_popular = ratings\
.groupBy("movieId")\
.agg(count("userId"))\
.withColumnRenamed("count(userId)", "num_ratings")\
.sort(desc("num_ratings"))
```

```
most_popular.show()
```

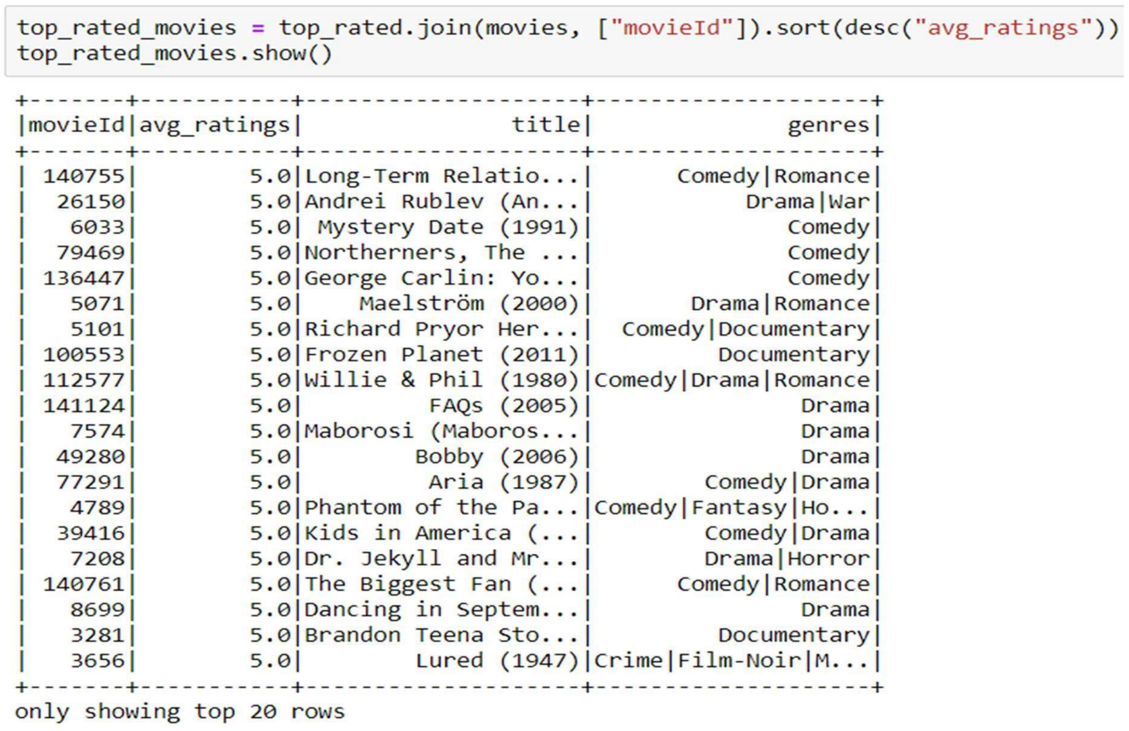
```
+-------+-----------+
|movieId|num_ratings|
+-------+-----------+
|    356|        341|
|    296|        324|
|    318|        311|
|    593|        304|
|    260|        291|
|    480|        274|
|   2571|        259|
|      1|        247|
|    527|        244|
|    589|        237|
|   1196|        234|
|    110|        228|
|   1270|        226|
|    608|        224|
|   1198|        220|
|   2858|        220|
|    780|        218|
|   1210|        217|
|    588|        215|
|    457|        213|
+-------+-----------+
only showing top 20 rows
```



```
most_popular_movies = most_popular.join(movies, ["movieId"])
most_popular_movies = most_popular_movies \
.sort(desc("num_ratings"))
most_popular_movies.show()
```

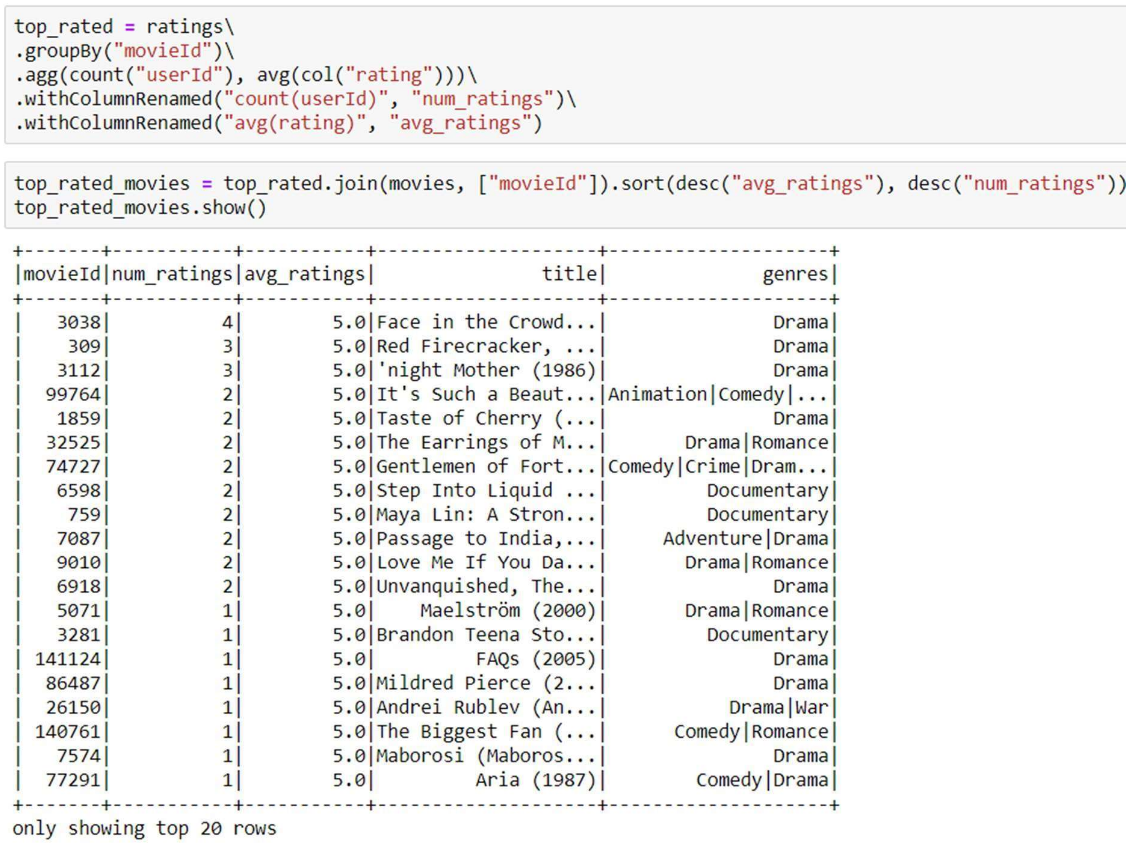
```
+-------+-----------+--------------------+--------------------+
|movieId|num_ratings|               title|              genres|
+-------+-----------+--------------------+--------------------+
|    356|        341| Forrest Gump (1994)|Comedy|Drama|Roma...|
|    296|        324| Pulp Fiction (1994)|Comedy|Crime|Dram...|
|    318|        311|Shawshank Redempt...|         Crime|Drama|
|    593|        304|Silence of the La...|Crime|Horror|Thri...|
|    260|        291|Star Wars: Episod...|Action|Adventure|...|
|    480|        274|Jurassic Park (1993)|Action|Adventure|...|
|   2571|        259|  Matrix, The (1999)|Action|Sci-Fi|Thr...|
|      1|        247|    Toy Story (1995)|Adventure|Animati...|
|    527|        244|Schindler's List ...|           Drama|War|
|    589|        237|Terminator 2: Jud...|       Action|Sci-Fi|
|   1196|        234|Star Wars: Episod...|Action|Adventure|...|
|    110|        228|   Braveheart (1995)|    Action|Drama|War|
|   1270|        226|Back to the Futur...|Adventure|Comedy|...|
|    608|        224|       Fargo (1996)|Comedy|Crime|Dram...|
|   1198|        220|Raiders of the Lo...|    Action|Adventure|
|   2858|        220|American Beauty (...|       Drama|Romance|
|    780|        218|Independence Day ...|Action|Adventure|...|
|   1210|        217|Star Wars: Episod...|Action|Adventure|...|
|    588|        215|      Aladdin (1992)|Adventure|Animati...|
|    457|        213|Fugitive, The (1993)|            Thriller|
+-------+-----------+--------------------+--------------------+
only showing top 20 rows
```

<u>Finding the average ratings of movies and sort them in the descending order</u> <u>of their average rating</u>.



```
top_rated_movies = top_rated.join(movies, ["movieId"]).sort(desc("avg_ratings"))
top_rated_movies.show()
```

```
+-------+-----------+--------------------+--------------------+
|movieId|avg_ratings|               title|              genres|
+-------+-----------+--------------------+--------------------+
| 140755|        5.0|Long-Term Relatio...|      Comedy|Romance|
|  26150|        5.0|Andrei Rublev (An...|           Drama|War|
|   6033|        5.0| Mystery Date (1991)|              Comedy|
|  79469|        5.0|Northerners, The ...|              Comedy|
| 136447|        5.0|George Carlin: Yo...|              Comedy|
|   5071|        5.0|    Maelström (2000)|       Drama|Romance|
|   5101|        5.0|Richard Pryor Her...|  Comedy|Documentary|
| 100553|        5.0|Frozen Planet (2011)|         Documentary|
| 112577|        5.0|Willie & Phil (1980)|Comedy|Drama|Romance|
| 141124|        5.0|         FAQs (2005)|               Drama|
|   7574|        5.0|Maborosi (Maboros...|               Drama|
|  49280|        5.0|        Bobby (2006)|               Drama|
|  77291|        5.0|         Aria (1987)|        Comedy|Drama|
|   4789|        5.0|Phantom of the Pa...|Comedy|Fantasy|Ho...|
|  39416|        5.0|Kids in America (...|        Comedy|Drama|
|   7208|        5.0|Dr. Jekyll and Mr...|        Drama|Horror|
| 140761|        5.0|The Biggest Fan (...|      Comedy|Romance|
|   8699|        5.0|Dancing in Septem...|               Drama|
|   3281|        5.0|Brandon Teena Sto...|         Documentary|
|   3656|        5.0|        Lured (1947)|Crime|Film-Noir|M...|
+-------+-----------+--------------------+--------------------+
only showing top 20 rows
```



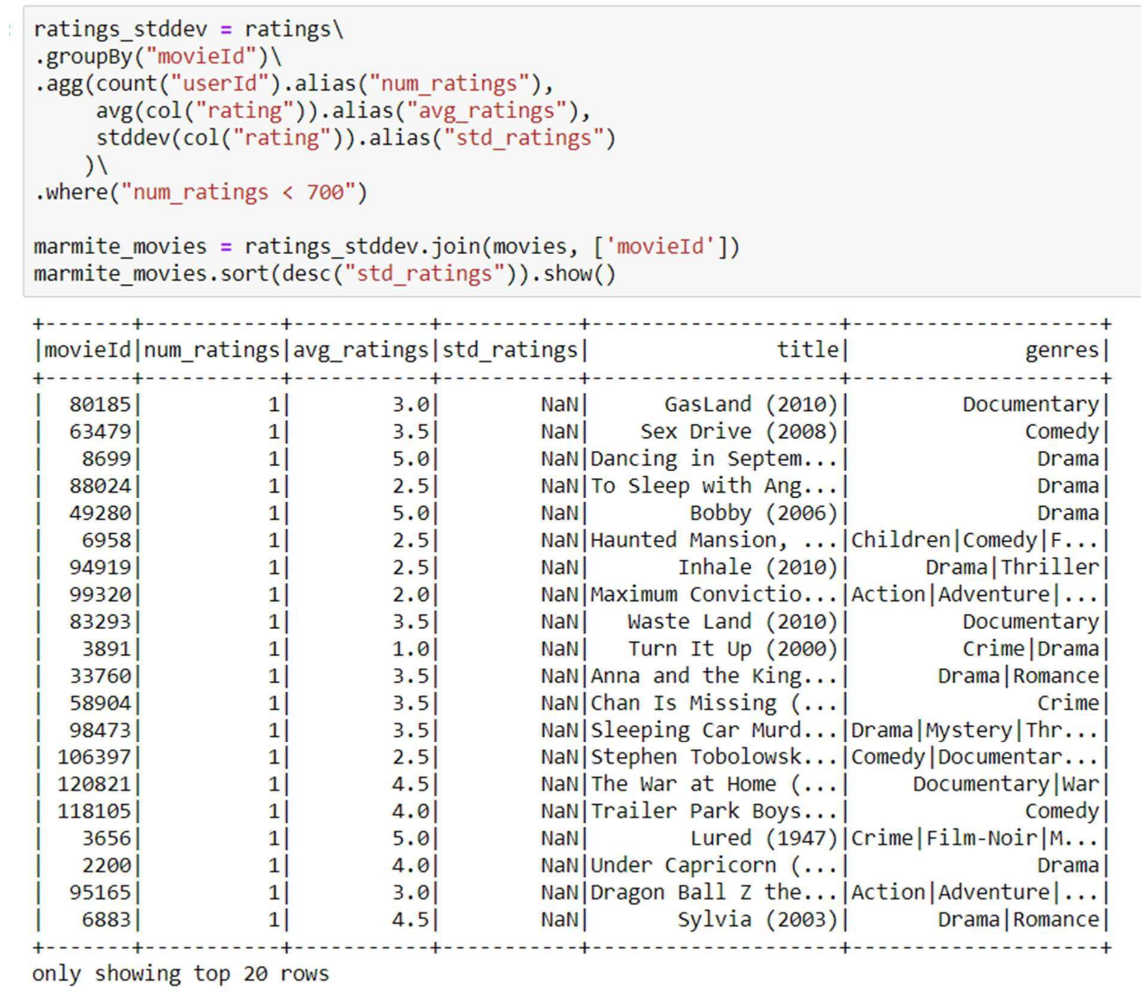
```
top_rated = ratings\
.groupBy("movieId")\
.agg(count("userId"), avg(col("rating")))\
.withColumnRenamed("count(userId)", "num_ratings")\
.withColumnRenamed("avg(rating)", "avg_ratings")
```

```
top_rated_movies = top_rated.join(movies, ["movieId"]).sort(desc("avg_ratings"), desc("num_ratings"))
top_rated_movies.show()
```

```
+-------+-----------+-----------+--------------------+--------------------+
|movieId|num_ratings|avg_ratings|               title|              genres|
+-------+-----------+-----------+--------------------+--------------------+
|   3038|          4|        5.0|Face in the Crowd...|               Drama|
|    309|          3|        5.0|Red Firecracker, ...|               Drama|
|   3112|          3|        5.0|'night Mother (1986)|               Drama|
|  99764|          2|        5.0|It's Such a Beaut...|Animation|Comedy|...|
|   1859|          2|        5.0|Taste of Cherry (...|               Drama|
|  32525|          2|        5.0|The Earrings of M...|       Drama|Romance|
|  74727|          2|        5.0|Gentlemen of Fort...|Comedy|Crime|Dram...|
|   6598|          2|        5.0|Step Into Liquid ...|         Documentary|
|    759|          2|        5.0|Maya Lin: A Stron...|         Documentary|
|   7087|          2|        5.0|Passage to India,...|     Adventure|Drama|
|   9010|          2|        5.0|Love Me If You Da...|       Drama|Romance|
|   6918|          2|        5.0|Unvanquished, The...|               Drama|
|   5071|          1|        5.0|    Maelström (2000)|       Drama|Romance|
|   3281|          1|        5.0|Brandon Teena Sto...|         Documentary|
| 141124|          1|        5.0|         FAQs (2005)|               Drama|
|  86487|          1|        5.0|Mildred Pierce (2...|               Drama|
|  26150|          1|        5.0|Andrei Rublev (An...|           Drama|War|
| 140761|          1|        5.0|The Biggest Fan (...|      Comedy|Romance|
|   7574|          1|        5.0|Maborosi (Maboros...|               Drama|
|  77291|          1|        5.0|         Aria (1987)|        Comedy|Drama|
+-------+-----------+-----------+--------------------+--------------------+
only showing top 20 rows
```

MOST POLARIZING MOVIES

Marmite movies are those which people either love or hate. We can find these movies by looking for the ones which have the highest standard deviation in the ratings.



```
ratings_stddev = ratings\
.groupBy("movieId")\
.agg(count("userId").alias("num_ratings"),
     avg(col("rating")).alias("avg_ratings"),
     stddev(col("rating")).alias("std_ratings")
    )\
.where("num_ratings < 700")

marmite_movies = ratings_stddev.join(movies, ['movieId'])
marmite_movies.sort(desc("std_ratings")).show()
```

```
+-------+-----------+-----------+-----------+--------------------+--------------------+
|movieId|num_ratings|avg_ratings|std_ratings|               title|              genres|
+-------+-----------+-----------+-----------+--------------------+--------------------+
|  80185|          1|        3.0|        NaN|      GasLand (2010)|         Documentary|
|  63479|          1|        3.5|        NaN|    Sex Drive (2008)|              Comedy|
|   8699|          1|        5.0|        NaN|Dancing in Septem...|               Drama|
|  88024|          1|        2.5|        NaN|To Sleep with Ang...|               Drama|
|  49280|          1|        5.0|        NaN|        Bobby (2006)|               Drama|
|   6958|          1|        2.5|        NaN|Haunted Mansion, ...|Children|Comedy|F...|
|  94919|          1|        2.5|        NaN|       Inhale (2010)|      Drama|Thriller|
|  99320|          1|        2.0|        NaN|Maximum Convictio...|Action|Adventure|...|
|  83293|          1|        3.5|        NaN|   Waste Land (2010)|         Documentary|
|   3891|          1|        1.0|        NaN|   Turn It Up (2000)|         Crime|Drama|
|  33760|          1|        3.5|        NaN|Anna and the King...|       Drama|Romance|
|  58904|          1|        3.5|        NaN|Chan Is Missing (...|               Crime|
|  98473|          1|        3.5|        NaN|Sleeping Car Murd...|Drama|Mystery|Thr...|
| 106397|          1|        2.5|        NaN|Stephen Tobolowsk...|Comedy|Documentar...|
| 120821|          1|        4.5|        NaN|The War at Home (...|     Documentary|War|
| 118105|          1|        4.0|        NaN|Trailer Park Boys...|              Comedy|
|   3656|          1|        5.0|        NaN|        Lured (1947)|Crime|Film-Noir|M...|
|   2200|          1|        4.0|        NaN|Under Capricorn (...|               Drama|
|  95165|          1|        3.0|        NaN|Dragon Ball Z the...|Action|Adventure|...|
|   6883|          1|        4.5|        NaN|       Sylvia (2003)|       Drama|Romance|
+-------+-----------+-----------+-----------+--------------------+--------------------+
only showing top 20 rows
```

VISUALIZATIONS

Using Seaborn and Koalas for doing the visualizations because they convert the spark data objects to something remarkably similar to Pandas data frames making it very easy to operate and plot graphs.

```
#installing koalas is nedded for visualizations.
!pip install koalas
```

```
Requirement already satisfied: koalas in c:\users\sindhura\anaconda3\lib\site-packages (1.4.0)
Requirement already satisfied: pandas>=0.23.2 in c:\users\sindhura\anaconda3\lib\site-packages (from koalas) (0.25.1)
Requirement already satisfied: pyarrow>=0.10 in c:\users\sindhura\anaconda3\lib\site-packages (from koalas) (2.0.0)
Requirement already satisfied: numpy>=1.14 in c:\users\sindhura\anaconda3\lib\site-packages (from koalas) (1.16.5)
Requirement already satisfied: matplotlib>=3.0.0 in c:\users\sindhura\anaconda3\lib\site-packages (from koalas) (3.1.1)
Requirement already satisfied: pytz>=2017.2 in c:\users\sindhura\anaconda3\lib\site-packages (from pandas>=0.23.2->koalas) (201
9.3)
Requirement already satisfied: python-dateutil>=2.6.1 in c:\users\sindhura\anaconda3\lib\site-packages (from pandas>=0.23.2->ko
alas) (2.8.0)
Requirement already satisfied: cycler>=0.10 in c:\users\sindhura\anaconda3\lib\site-packages (from matplotlib>=3.0.0->koalas)
(0.10.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\sindhura\anaconda3\lib\site-packages (from matplotlib>=3.0.0->koal
as) (1.1.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in c:\users\sindhura\anaconda3\lib\site-packages (from
matplotlib>=3.0.0->koalas) (2.4.2)
Requirement already satisfied: six>=1.5 in c:\users\sindhura\anaconda3\lib\site-packages (from python-dateutil>=2.6.1->pandas>=
0.23.2->koalas) (1.12.0)
Requirement already satisfied: setuptools in c:\users\sindhura\anaconda3\lib\site-packages (from kiwisolver>=1.0.1->matplotlib>
=3.0.0->koalas) (41.4.0)
```

```
!pip install seaborn
```

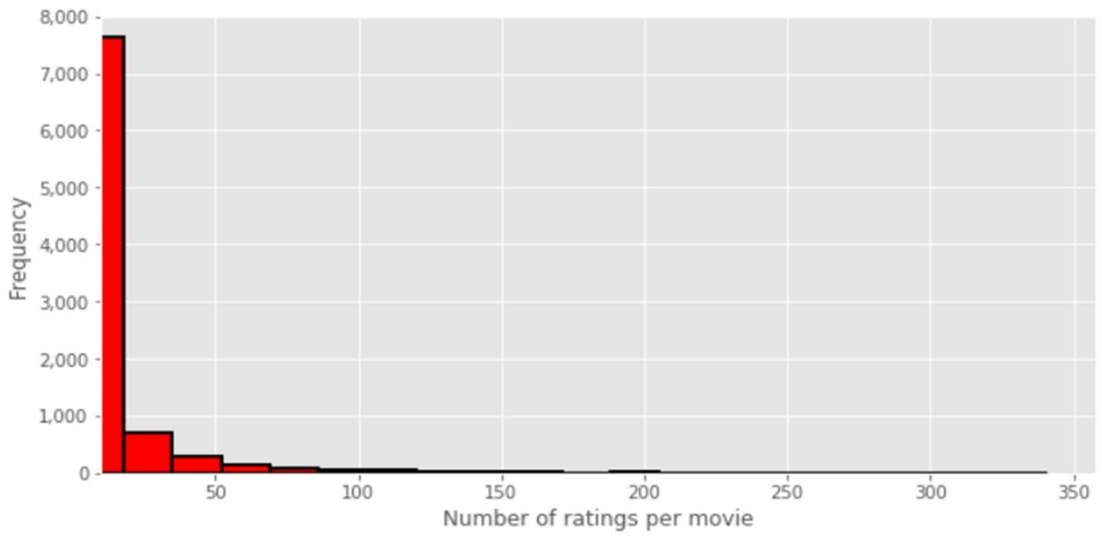
```
Requirement already satisfied: seaborn in c:\users\sindhura\anaconda3\lib\site-packages (0.9.0)
Requirement already satisfied: numpy>=1.9.3 in c:\users\sindhura\anaconda3\lib\site-packages (from seaborn) (1.16.5)
Requirement already satisfied: matplotlib>=1.4.3 in c:\users\sindhura\anaconda3\lib\site-packages (from seaborn) (3.1.1)
Requirement already satisfied: scipy>=0.14.0 in c:\users\sindhura\anaconda3\lib\site-packages (from seaborn) (1.3.1)
Requirement already satisfied: pandas>=0.15.2 in c:\users\sindhura\anaconda3\lib\site-packages (from seaborn) (0.25.1)
Requirement already satisfied: cycler>=0.10 in c:\users\sindhura\anaconda3\lib\site-packages (from matplotlib>=1.4.3->seaborn)
(0.10.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\sindhura\anaconda3\lib\site-packages (from matplotlib>=1.4.3->seab
orn) (1.1.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in c:\users\sindhura\anaconda3\lib\site-packages (from
matplotlib>=1.4.3->seaborn) (2.4.2)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\sindhura\anaconda3\lib\site-packages (from matplotlib>=1.4.3->s
eaborn) (2.8.0)
Requirement already satisfied: pytz>=2017.2 in c:\users\sindhura\anaconda3\lib\site-packages (from pandas>=0.15.2->seaborn) (20
19.3)
Requirement already satisfied: six in c:\users\sindhura\anaconda3\lib\site-packages (from cycler>=0.10->matplotlib>=1.4.3->seab
orn) (1.12.0)
Requirement already satisfied: setuptools in c:\users\sindhura\anaconda3\lib\site-packages (from kiwisolver>=1.0.1->matplotlib>
=1.4.3->seaborn) (41.4.0)
```

Visualizing the number of ratings given by users. We convert the spark data object to a Koalas data frame and then use this matplotlib package to plot the graph.

Number of ratings per movie

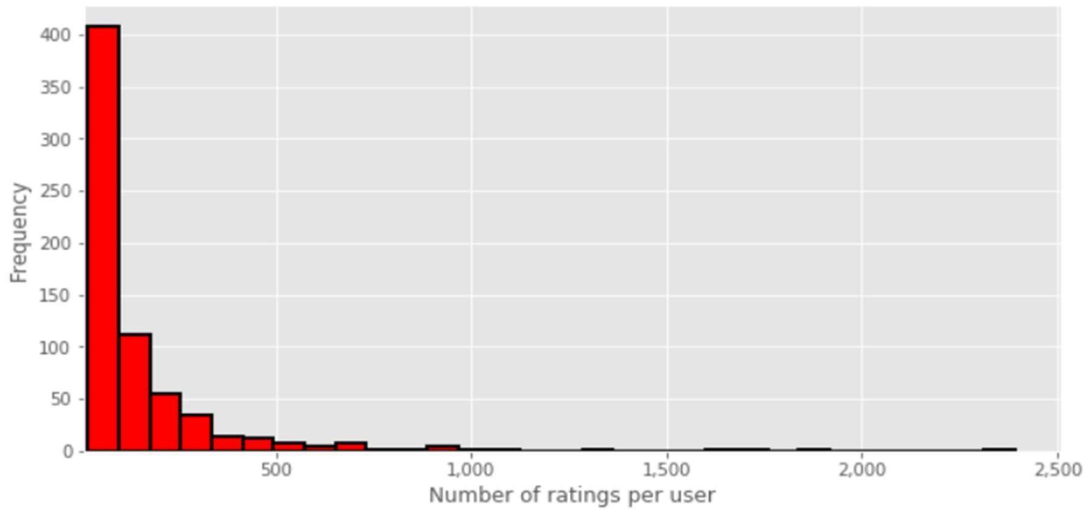
```
ks.set_option('compute.default_index_type', 'sequence')
ks.set_option('compute.ops_on_diff_frames', True)
dfRatingsKdf = ratings.to_koalas()
```

```
f, ax = plt.subplots(figsize=(10,5))
userRatingGroup = dfRatingsKdf.groupby("movieId")["rating"].count()
userRatingGroup.hist(bins=20, color='red', edgecolor='black',
                     linewidth=2, alpha=1, ax=ax)
ax.set_xlabel('Number of ratings per movie')
ax.set_xlim(10.10)
ax.set_xticklabels(['{:,}'.format(int(x)) for x in ax.get_xticks().tolist()])
ax.set_yticklabels(['{:,}'.format(int(y)) for y in ax.get_yticks().tolist()])
plt.show()
```



Number of ratings per user

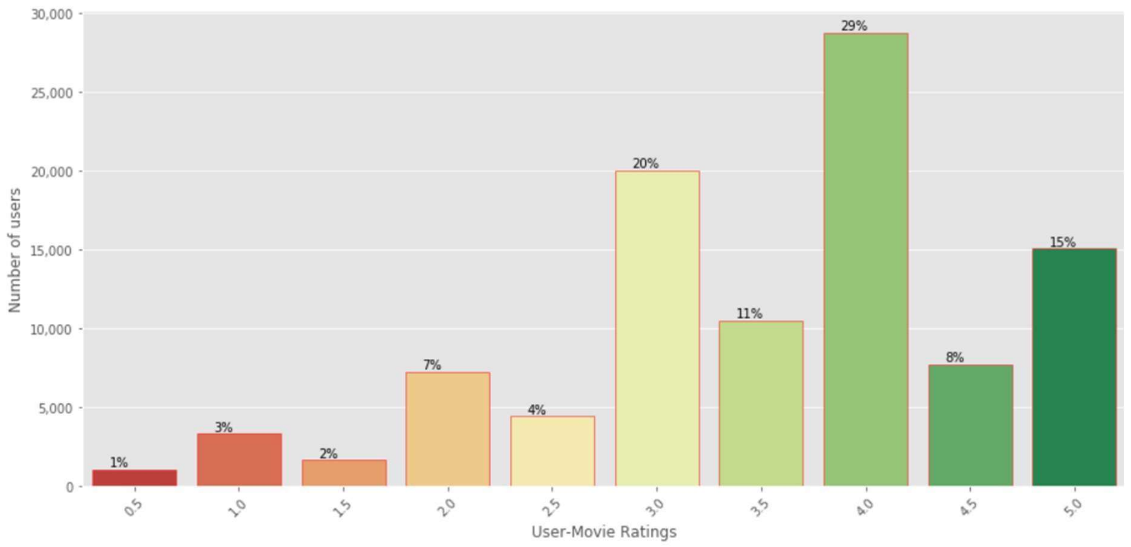
```
f, ax = plt.subplots(figsize=(10,5))
userRatingGroup = dfRatingsKdf.groupby("userId")["rating"].count()
userRatingGroup.hist(bins=30, color='red', edgecolor='black',
                    linewidth=2, alpha=1, ax=ax)
ax.set_xlabel('Number of ratings per user')
ax.set_xlim(10.10)
ax.set_xticklabels(['{:,}'.format(int(x)) for x in ax.get_xticks().tolist()])
ax.set_yticklabels(['{:,}'.format(int(y)) for y in ax.get_yticks().tolist()])
plt.show()
```



USER RATING ON MOVIES

```python
movieRatingsDistGroup = dfRatingsKdf['rating'].value_counts() \
.sort_index() \
.reset_index() \
.to_pandas()
```

```python
# Create MatpLot&ib Figure
fig, ax = plt.subplots(figsize=(15,7))

#seaborn barplot
sns.barplot(data=movieRatingsDistGroup, x='index', y='rating',
            palette='RdYlGn', edgecolor="red", ax=ax)

#setting x and y axis
ax.set_xlabel("User-Movie Ratings")
ax.set_ylabel("Number of users")
ax.xaxis.set_tick_params(rotation=45)

#thousand seperator on yaxis
ax.set_yticklabels(['{:,}'.format(int(x)) for x in ax.get_yticks().tolist()])

#adding percentage on each bar
total = float(movieRatingsDistGroup['rating'].sum())
for p in ax.patches:
    height = p.get_height()
    ax.text(p.get_x()+p.get_width()/5,
    height+200,
    '{0:.0%}'.format(height/total),
    ha="left")

#display the plot
plt.show()
```


Number of users
User-Movie Ratings