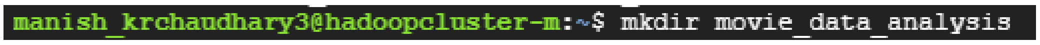
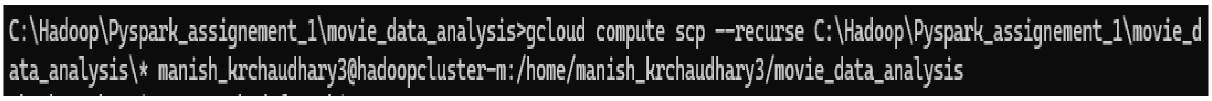
Movie Data Analysis

1) Setuped the hadoop cluster using GCP.
2) Created a folder in local VM of gcp ssh using command



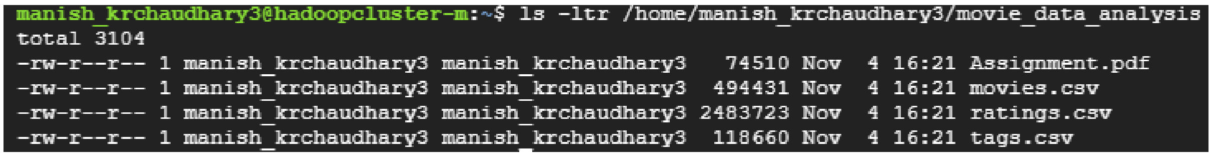
```
manish_krchaudhary3@hadoopcluster-m:~$ mkdir movie_data_analysis
```

3) Now lets move the data from our local space to VM local



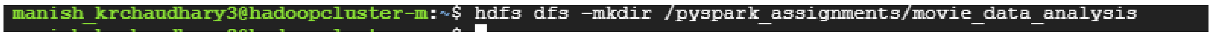
```
C:\Hadoop\Pyspark_assignement_1\movie_data_analysis>gcloud compute scp --recurse C:\Hadoop\Pyspark_assignement_1\movie_d
ata_analysis\* manish_krchaudhary3@hadoopcluster-m:/home/manish_krchaudhary3/movie_data_analysis
```

4) Checking whether data has been copied to local VM or not



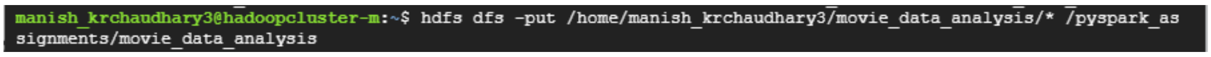
```
manish_krchaudhary3@hadoopcluster-m:~$ ls -ltr /home/manish_krchaudhary3/movie_data_analysis
total 3104
-rw-r--r-- 1 manish_krchaudhary3 manish_krchaudhary3   74510 Nov  4 16:21 Assignment.pdf
-rw-r--r-- 1 manish_krchaudhary3 manish_krchaudhary3  494431 Nov  4 16:21 movies.csv
-rw-r--r-- 1 manish_krchaudhary3 manish_krchaudhary3 2483723 Nov  4 16:21 ratings.csv
-rw-r--r-- 1 manish_krchaudhary3 manish_krchaudhary3  118660 Nov  4 16:21 tags.csv
```

5) Creating a folder in hdfs location as the data need to be copied there so that we can ready with help of spark.



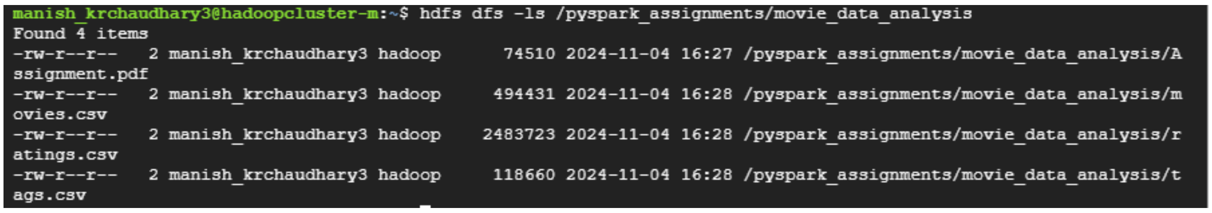
```
manish_krchaudhary3@hadoopcluster-m:~$ hdfs dfs -mkdir /pyspark_assignments/movie_data_analysis
```

6) Moving data from local VM to hdfs location



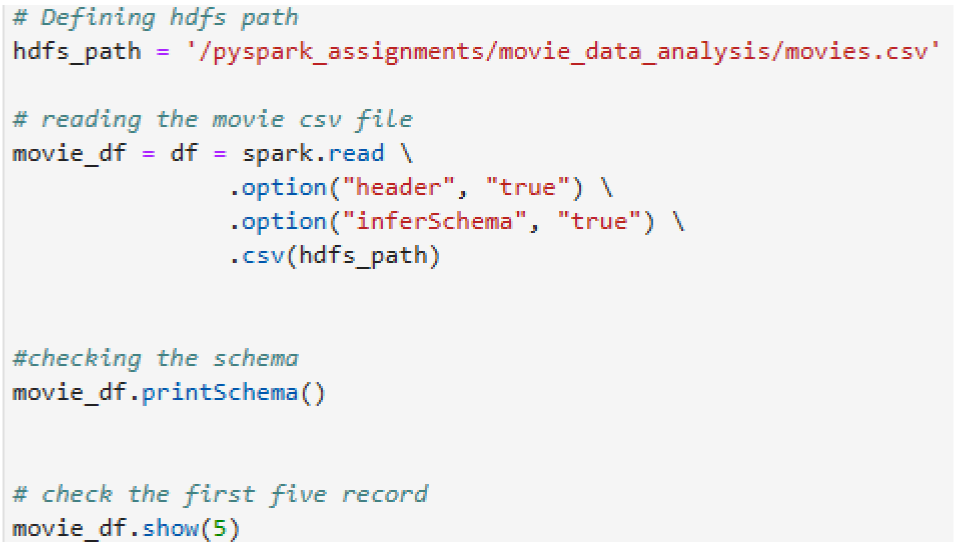
```
manish_krchaudhary3@hadoopcluster-m:~$ hdfs dfs -put /home/manish_krchaudhary3/movie_data_analysis/* /pyspark_as
signments/movie_data_analysis
```

7) Checking if data transferred to hdfs location or not



```
manish_krchaudhary3@hadoopcluster-m:~$ hdfs dfs -ls /pyspark_assignments/movie_data_analysis
Found 4 items
-rw-r--r--   2 manish_krchaudhary3 hadoop      74510 2024-11-04 16:27 /pyspark_assignments/movie_data_analysis/A
ssignment.pdf
-rw-r--r--   2 manish_krchaudhary3 hadoop     494431 2024-11-04 16:28 /pyspark_assignments/movie_data_analysis/m
ovies.csv
-rw-r--r--   2 manish_krchaudhary3 hadoop    2483723 2024-11-04 16:28 /pyspark_assignments/movie_data_analysis/r
atings.csv
-rw-r--r--   2 manish_krchaudhary3 hadoop     118660 2024-11-04 16:28 /pyspark_assignments/movie_data_analysis/t
ags.csv
```

8) Now running the jupyterlab and loading the datasets from hdfs location
   a) Movies data



```python
# Defining hdfs path
hdfs_path = '/pyspark_assignments/movie_data_analysis/movies.csv'

# reading the movie csv file
movie_df = df = spark.read \
               .option("header", "true") \
               .option("inferSchema", "true") \
               .csv(hdfs_path)


#checking the schema
movie_df.printSchema()


# check the first five record
movie_df.show(5)
```

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

[Stage 2:>
+-------+--------------------+--------------------+
|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|
+-------+--------------------+--------------------+
only showing top 5 rows
```

b) Ratings data

```python
# Defining hdfs path
hdfs_path = '/pyspark_assignments/movie_data_analysis/ratings.csv'

# reading the movie csv file
rating_df = df = spark.read \
                .option("header", "true") \
                .option("inferSchema", "true") \
                .csv(hdfs_path)


#checking the schema
rating_df.printSchema()


# check the first five record
rating_df.show(5,False)
```

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

+------+-------+------+---------+
|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|
+------+-------+------+---------+
only showing top 5 rows
```

c) Tags data

```
# Defining hdfs path
hdfs_path = '/pyspark_assignments/movie_data_analysis/tags.csv'

# reading the movie csv file
tag_df = df = spark.read \
               .option("header", "true") \
               .option("inferSchema", "true") \
               .csv(hdfs_path)


#checking the schema
tag_df.printSchema()


# check the first five record
tag_df.show(5,False)
```

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

+------+-------+---------------+----------+
|userId|movieId|tag            |timestamp |
+------+-------+---------------+----------+
|2     |60756  |funny          |1445714994|
|2     |60756  |Highly quotable|1445714996|
|2     |60756  |will ferrell   |1445714992|
|2     |89774  |Boxing story   |1445715207|
|2     |89774  |MMA            |1445715200|
+------+-------+---------------+----------+
```

9) Solving the questions now
   a) Show the aggregated number of ratings per year

Issue: **While I uploaded the datasets, the timestamp was in integer format thus I had convert it back to normal datetime in pyspark.**

```
#converting the timestamp numeric format into normal timestamp format
from pyspark.sql.functions import from_unixtime
rating_df = rating_df.withColumn("timestamp", from_unixtime("timestamp"))

#extracting year and month from timestamp column
rating_df=rating_df.withColumn("year",year("timestamp"))\
                   .withColumn("month",month("timestamp"))
```

After this I worked for the required output which was based out of basic grouping.

```
#grouping to get the result
result_1=rating_df\
          .groupBy('year')\
          .agg(count('rating').alias('no_of_rating'))\
          .orderBy('year')
```

Output:

```
result_1.show(5)
```

```
+----+------------+
|year|no_of_rating|
+----+------------+
|1996|        6040|
|1997|        1916|
|1998|         507|
|1999|        2439|
|2000|       10061|
+----+------------+
only showing top 5 rows
```

b) Show the average monthly number of ratings

```
# b. Show the average monthly number of ratings
result_2_intermediate=rating_df\
                            .groupBy(['year','month'])\
                            .agg(round(count('rating'),2).alias('no_of_rating')).orderBy(['year','month'])
result_2=result_2_intermediate\
                  .groupBy('month').agg(round(avg('no_of_rating'),2).alias('avg_no_of_rating'))\
                  .orderBy('month')
```

Output:

```
result_2.show()
```

```
[Stage 12:>
+-----+----------------+
|month|avg_no_of_rating|
+-----+----------------+
|    1|          394.73|
|    2|          347.05|
|    3|          386.09|
|    4|          367.95|
|    5|          473.17|
|    6|           383.7|
|    7|          302.17|
|    8|          412.45|
|    9|          386.82|
|   10|          324.91|
|   11|          439.82|
|   12|          311.09|
+-----+----------------+
```

c) Show the rating levels distribution

```
# c. Show the rating levels distribution
result_3=rating_df\
            .groupBy('rating')\
            .agg(count('rating').alias('no_of_rating')
                 ,countDistinct('userid').alias('ratedby_users_count')
                 ,countDistinct('movieid').alias('ratedfor_movies_count'))\
            .orderBy('rating')
```

Output:

```
result_3.show()
```

```
[Stage 30:>
+------+------------+------------------+--------------------+
|rating|no_of_rating|ratedby_users_count|ratedfor_movies_count|
+------+------------+------------------+--------------------+
|   0.5|        1370|               179|                1066|
|   1.0|        2811|               351|                1726|
|   1.5|        1791|               191|                1386|
|   2.0|        7551|               491|                3339|
|   2.5|        5550|               291|                2925|
|   3.0|       20047|               585|                4986|
|   3.5|       13136|               355|                4216|
|   4.0|       26818|               606|                5109|
|   4.5|        8551|               354|                2710|
|   5.0|       13211|               573|                2954|
+------+------------+------------------+--------------------+
```

d) Show the 18 movies that are tagged but not rated

```
result_4=movie_rating_tag_df\
                .where((movie_rating_tag_df['tag_userid'].isNotNull()) \
                     & (movie_rating_tag_df['rating_userid'].isNull()))[['movieid','title']]\
                .distinct()
```

Output:

```
result_4.show(100,False)
```

```
+-------+-----------------------------------------+
|movieid|title                                    |
+-------+-----------------------------------------+
|32160  |Twentieth Century (1934)                 |
|7792   |Parallax View, The (1974)                |
|32371  |Call Northside 777 (1948)                |
|1076   |Innocents, The (1961)                    |
|6849   |Scrooge (1970)                           |
|25855  |Roaring Twenties, The (1939)             |
|8765   |This Gun for Hire (1942)                 |
|34482  |Browning Version, The (1951)             |
|7020   |Proof (1991)                             |
|26085  |Mutiny on the Bounty (1962)              |
|4194   |I Know Where I'm Going! (1945)           |
|3456   |Color of Paradise, The (Rang-e khoda) (1999)|
|5721   |Chosen, The (1981)                       |
|85565  |Chalet Girl (2011)                       |
|30892  |In the Realms of the Unreal (2004)       |
|2939   |Niagara (1953)                           |
|3338   |For All Mankind (1989)                   |
|6668   |Road Home, The (Wo de fu qin mu qin) (1999) |
+-------+-----------------------------------------+
```

e) Show the movies that have rating but no tag

```
# e. Show the movies that have rating but no tag

#so this is just opposite of above as we already have joined df we directly go with result part
result_5=movie_rating_tag_df\
                    .where((movie_rating_tag_df['tag_userid'].isNull()) \
                          & (movie_rating_tag_df['rating_userid'].isNotNull()))[['movieid','title']]\
                    .distinct()
```

Output:

```
result_5.show(5,False)
```

```
+-------+--------------------------+
|movieid|title                     |
+-------+--------------------------+
|141688 |Legend (2015)             |
|4152   |Vatel (2000)              |
|180297 |The Disaster Artist (2017)|
|2907   |Superstar (1999)          |
|2824   |On the Ropes (1999)       |
+-------+--------------------------+
only showing top 5 rows
```

As there are many records displaying top 5 only.
Let's see how many movies there are.

```
result_5.count()
```

```
8170
```

f) Focusing on the rated untagged movies with more than 30 user ratings, show the top 10 movies in terms of average rating and number of ratings

Explanation:

a) At first I have just filtered out data based on condition that is rated and untagged.
b) After that grouped on the basis of movieid and title , to get the average rating, no of users rated, no of ratings.
c) Again filtered on top of it to get those records only where no of users who rated are more than 30.
d) Finally did ordering on the basis of average rating and no of rating and used limit to to get the top 10 records only.

```
# f. Focusing on the rated untagged movies with more than 30 user ratings,
# show the top 10 movies in terms of average rating and number of
# ratings

result_6=movie_rating_tag_df\
                    .where((movie_rating_tag_df['tag_userid'].isNull()) \
                           & (movie_rating_tag_df['rating_userid'].isNotNull()))\
                    .groupBy(['movieid','title']).agg(countDistinct('rating_userid').alias('no_of_users_rated')
                                                   ,avg('rating').alias('average_rating')
                                                   ,count('rating_userid').alias('no_of_ratings'))\
                    .where(col('no_of_users_rated')>30)\
                    .orderBy(col('average_rating').desc(),col('no_of_ratings').desc())\
                    .limit(10)
```

Output:

```
result_6.show()
```

```
[Stage 102:>                                                    (0 + 1) / 1]
+-------+--------------------+-----------------+------------------+-------------+
|movieid|               title|no_of_users_rated|    average_rating|no_of_ratings|
+-------+--------------------+-----------------+------------------+-------------+
|   3275|Boondock Saints, ...|               43| 4.22093023255814|           43|
|   1199|       Brazil (1985)|               59|4.177966101694915|           59|
|   1172|Cinema Paradiso (...|               34|4.161764705882353|           34|
|   4011|       Snatch (2000)|               93|4.155913978494624|           93|
|   3681|For a Few Dollars...|               33|4.151515151515151|           33|
|  44555|Lives of Others, ...|               34|4.117647058823529|           34|
|  78499|    Toy Story 3 (2010)|               55|4.109090909090909|           55|
|   1673| Boogie Nights (1997)|               39|4.076923076923077|           39|
|   2858|American Beauty (...|              204|4.056372549019608|          204|
|   2542|Lock, Stock & Two...|               67|4.052238805970149|           67|
+-------+--------------------+-----------------+------------------+-------------+
```

g) What is the average number of tags per movie in tagsDF? And the average number of tags per user? How does it compare with the average number of tags a user assigns to a movie?

```
# g. What is the average number of tags per movie in tagsDF? And the
# average number of tags per user? How does it compare with the
# average number of tags a user assigns to a movie?
result_7_1=tag_df\
            .groupBy('movieid')\
            .agg(count('tag').alias('no_of_tags'))\
            .agg(avg('no_of_tags').alias('avg_no_of_tags'))
result_7_2=tag_df\
            .groupBy('userid')\
            .agg(count('tag').alias('no_of_tags'))\
            .agg(avg('no_of_tags').alias('avg_no_of_tags'))

result_7_3=tag_df\
            .groupBy(['movieid','userid'])\
            .agg(count('tag').alias('no_of_tags'))\
            .agg(avg('no_of_tags').alias('avg_no_of_tags'))
```

Output:

```
result_7_1.show()
result_7_2.show()
result_7_3.show()
```

```
+------------------+
|     avg_no_of_tags|
+------------------+
|2.342875318066157 7|
+------------------+

+--------------+
|avg_no_of_tags|
+--------------+
|          63.5|
+--------------+

+-----------------+
|   avg_no_of_tags|
+-----------------+
|2.074929577464789|
+-----------------+
```

Inferences: From the above result, we can see that the average number of tags per movie is around 2.34, meaning that each movie receives approximately 2.34 tags. Similarly, the average number of tags per user is 63.5, which indicates that a user provides an average of 63.5 tags. However, when we look at the average number of tags per movie and user combination, we see it is around 2. This suggests that, in general, each movie has almost 2 tags. The high average of 63.5 tags per user is misleading because it likely reflects a small number of users who tag frequently, thereby skewing the average. Therefore, the third average indicates that a user typically provides around 2 tags per movie. When we aggregate across all users, this discrepancy becomes apparent, as the same user tags multiple movies.

h) Identify the users that tagged movies without rating them

```
# h. Identify the users that tagged movies without rating them

#Lets join tag_df with rating_df
tag_rating_df=tag_df.alias('t').join(rating_df.alias('r'),col('t.userid')==col('r.userid'),'left')\
                                .select('t.userid'
                                        ,'t.tag'
                                        ,col('t.timestamp').alias('tag_date')
                                        ,col('r.userid').alias('rating_userid')
                                        ,'r.rating'
                                        ,col('r.timestamp').alias('rating_date'))
```

```
#now lets final go with the required output

result_8=tag_rating_df\
            .where(tag_rating_df['rating_userid'].isNull())\
            .select('userid')
```

Output:

```
result_8.show()
```

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

i) What is the average number of ratings per user in ratings DF? And the average number of ratings per movie?

```
# i. What is the average number of ratings per user in ratings DF? And the
# average number of ratings per movie?

result_9_1=rating_df.groupBy('movieid')\
         .agg(count('movieid').alias('no_of_ratings'))\
         .agg(round(avg('no_of_ratings'),2).alias('avg_no_of_rating_per_movie'))


result_9_2=rating_df.groupBy('userid')\
         .agg(count('userid').alias('no_of_ratings'))\
         .agg(round(avg('no_of_ratings'),2).alias('avg_no_of_rating_per_user'))
```

Output:

```
result_9_1.show()
result_9_2.show()
```

```
+--------------------------+
|avg_no_of_rating_per_movie|
+--------------------------+
|                     10.37|
+--------------------------+

+-------------------------+
|avg_no_of_rating_per_user|
+-------------------------+
|                    165.3|
+-------------------------+
```

j) What is the predominant (frequency based) genre per rating level?
Here, firstly calculated the genre count for each rating and after that ranked them in descending order of genre count and took the rank with 1.

```
# j. What is the predominant (frequency based) genre per rating level?

#lets join movie and rating table and get frequecny count
movie_rating_df=movie_df.alias('m')\
                         .join(rating_df.alias('r'),col('m.movieid')==col('r.movieid'),'inner')\
                         .groupBy(['rating','genres']).agg(count('genres').alias('genre_count'))
```

```
#now lets rank the frequency for each rating so that we can get predominant genres
from pyspark.sql.window import Window
from pyspark.sql import functions as F

#first lets define the window_spec
window_spec=Window.partitionBy('rating').orderBy(desc('genre_count'))

result_10=movie_rating_df.withColumn('rank',row_number().over(window_spec)).filter(col('rank')==1)
```

Output:

```
result_10.show(100,False)
```

```
+------+------+-----------+----+
|rating|genres|genre_count|rank|
+------+------+-----------+----+
|0.5   |Comedy|136        |1   |
|1.0   |Comedy|348        |1   |
|1.5   |Comedy|256        |1   |
|2.0   |Comedy|828        |1   |
|2.5   |Comedy|515        |1   |
|3.0   |Comedy|1614       |1   |
|3.5   |Comedy|854        |1   |
|4.0   |Drama |2055       |1   |
|4.5   |Drama |593        |1   |
|5.0   |Drama |895        |1   |
+------+------+-----------+----+
```

k) What is the predominant tag per genre ?

```
# k. What is the predominant tag per genre?

#first lets join the movie and tag df and get the count of genre

movie_tag_df=movie_df.alias('m')\
                         .join(tag_df.alias('t'),col('m.movieid')==col('t.movieid'),'inner')\
                         .groupBy(['tag','genres']).agg(count('genres').alias('genre_count'))
```

```
#now lets rank

#first lets define the window_spec
window_spec=Window.partitionBy('tag').orderBy(desc('genre_count'))

result_11=movie_tag_df.withColumn('rank',row_number().over(window_spec)).filter(col('rank')==1)
```

Output:

```
result_11.show()
```

```
+--------------------+--------------------+
|                 tag|              genres|
+--------------------+--------------------+
|          """artsy"""|Action|Horror|Sci...|
|06 Oscar Nominate...|Adventure|Animati...|
|               1900s|             Musical|
|               1920s|        Comedy|Drama|
|               1950s|Horror|Sci-Fi|Thr...|
|               1960s|               Drama|
|               1970s|       Drama|Romance|
|               1980s|Comedy|Drama|Romance|
|               1990s|Comedy|Crime|Dram...|
|           2001-like|Drama|Mystery|Sci...|
|        2D animation|Animation|Comedy|...|
|                70mm|Comedy|Drama|Musical|
|                80's|              Comedy|
|                AIDs|               Drama|
|            AS Byatt|       Drama|Romance|
|             AWESOME|Comedy|Crime|Dram...|
|             Aardman|Adventure|Animati...|
|Academy award (Be...|   Drama|Romance|War|
|              Action|Action|Adventure|...|
|        Adam Sandler|              Comedy|
+--------------------+--------------------+
```

I) What are the most predominant (popularity based) movies?

Here, I have tried to find the most predominant movie based on the highest average rating, no of ratings and no of tags the movie has. And as a result I am showing the top 10 movies.

```
# l. What are the most predominant (popularity based) movies?

intermediate_result=movie_rating_tag_df.groupBy(['movieid','title'])\
                    .agg(countDistinct('rating_userid').alias('no_of_ratings')\
                        ,avg('rating').alias('avg_rating')\
                        ,countDistinct('tag_userid').alias('no_of_tags'))\
```

```
#now lets rnak based on rating,no_of_rating and no_of_tags

window_spec=Window.orderBy(desc('avg_rating'),desc('no_of_ratings'),desc('no_of_tags'))

result_12=intermediate_result.withColumn('rank',row_number().over(window_spec)).filter(col('rank')<=10)
```

Output:

```
+-------+--------------------+-------------+----------+----------+----+
|movieid|               title|no_of_ratings|avg_rating|no_of_tags|rank|
+-------+--------------------+-------------+----------+----------+----+
|   6818|Come and See (Idi...|            2|       5.0|         1|   1|
|     99|Heidi Fleiss: Hol...|            2|       5.0|         0|   2|
|     53|     Lamerica (1994)|            2|       5.0|         0|   3|
|  78836|Enter the Void (2...|            2|       5.0|         0|   4|
|   6442| Belle époque (1992)|            2|       5.0|         0|   5|
|   1151| Lesson Faust (1994)|            2|       5.0|         0|   6|
|   3473|Jonah Who Will Be...|            2|       5.0|         0|   7|
|   4495|Crossing Delancey...|            1|       5.0|         1|   8|
|   8580|Into the Woods (1...|            1|       5.0|         1|   9|
|   5088|Going Places (Val...|            1|       5.0|         1|  10|
+-------+--------------------+-------------+----------+----------+----+
```

m) Top 10 movies in terms of average rating (provided more than 30 users
# reviewed them)

```
# m. Top 10 movies in terms of average rating (provided more than 30 users
# reviewed them)
intermediate_result_13=movie_rating_tag_df.groupBy(['movieid','title'])\
                    .agg(countDistinct('rating_userid').alias('no_of_ratings')\
                        ,avg('rating').alias('avg_rating'))\
                    .where(col('no_of_ratings')>30)
```

```
#now rank

window_spec=Window.orderBy(desc('avg_rating'))

result_13=intermediate_result_13.withColumn('rank',row_number().over(window_spec)).filter(col('rank')<=10)
```

Output:

```
+-------+--------------------+-------------+-----------------+----+
|movieid|               title|no_of_ratings|       avg_rating|rank|
+-------+--------------------+-------------+-----------------+----+
|    318|Shawshank Redempt...|          317|4.429022082018927|   1|
|   1204|Lawrence of Arabi...|           45|              4.3|   2|
|    858|Godfather, The (1...|          192|        4.2890625|   3|
|   2959|   Fight Club (1999)|          218|4.272935779816514|   4|
|   1276|Cool Hand Luke (1...|           57|4.271929824561403|   5|
|    750|Dr. Strangelove o...|           97|4.268041237113402|   6|
|    904|   Rear Window (1954)|          84|4.261904761904762|   7|
|   1221|Godfather: Part I...|          129| 4.25968992248062|   8|
|  48516|Departed, The (2006)|          107|4.252336448598131|   9|
|   1213|   Goodfellas (1990)|          126|             4.25|  10|
+-------+--------------------+-------------+-----------------+----+
```