Name: Miral Kunapara
Email: miralkunapara2003@gmail.com

Spark Assignment-2(Movie Data Analysis) Documentation

This file details and describes all the attached files for the Spark Assignment-2

Tools Used:

1. Python3 – Microsoft VScode
2. Apache Spark (On GCP Cluster)
3. GCP services – DataProc/GCS
4. Jupyter Lab
5. Apache Hive

Files Attached:

1. Spark_Ass2.pdf – This file
2. Spark_MovieRating.– Pyspark File that details all the analysis

Process and File Descriptions:

Step 1:

I placed the three csv files in the HDFS location and ingested them into their respective spark dataframes. The three frames being

- Movies
- Ratings
- Tags

[1]:
```
# Reading movies data

hdfs_path = '/practice/movies.csv'
df_movies = spark.read.format('csv').option('header','true').option('inferSchema','true').load(hdfs_path)

#print schema and sample data
df_movies.printSchema()
df_movies.show(5)
```

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

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

```
[13]: from pyspark.sql.types import *
from pyspark.sql.functions import from_unixtime, unix_timestamp, col

# Define correct schema based on csv structure
schema = StructType([
    StructField("userid",IntegerType(),True),
    StructField("movieid",IntegerType(),True),
    StructField("rating",FloatType(),True),
    StructField("timestamp",IntegerType(),True),
])

hdfs_path = '/practice/ratings.csv'

# reading csv file into  a Dataframe

df_ratings = spark.read.format('csv').option('header','true').option('inferSchema', 'false').schema(schema).load(hdfs_path)

# convert timestamp to Timestamptype

df_ratings = df_ratings.withColumn("timestamp",from_unixtime("timestamp").cast(TimestampType()))

#show the dataframe

df_ratings.show(5)
```

```
+------+-------+------+-------------------+
|userid|movieid|rating|          timestamp|
+------+-------+------+-------------------+
|     1|      1|   4.0|2000-07-30 18:45:03|
|     1|      3|   4.0|2000-07-30 18:20:47|
|     1|      6|   4.0|2000-07-30 18:37:04|
|     1|     47|   5.0|2000-07-30 19:03:35|
|     1|     50|   5.0|2000-07-30 18:48:51|
+------+-------+------+-------------------+
only showing top 5 rows
```

```
[15]: #Define correct schema based on csv structure

schema = StructType([
     StructField("userId",IntegerType(),True),
     StructField("movieId",IntegerType(),True),
     StructField("tag",StringType(),True),
     StructField("timestamp",IntegerType(),True),])


hdfs_path = '/practice/tags.csv'

#Read the csv file into a dataframe
df_tags = spark.read.format('csv').option('header','true').option('inferSchema','false').schema(schema).load(hdfs_path)

#convert timestamp to TimestampType

df_tags = df_tags.withColumn("timestamp",from_unixtime('timestamp').cast(TimestampType()))


#show the dataframe
df_tags.show(5)
```

```
+------+-------+---------------+-------------------+
|userId|movieId|            tag|          timestamp|
+------+-------+---------------+-------------------+
|     2|  60756|          funny|2015-10-24 19:29:54|
|     2|  60756|Highly quotable|2015-10-24 19:29:56|
|     2|  60756|   will ferrell|2015-10-24 19:29:52|
|     2|  89774|   Boxing story|2015-10-24 19:33:27|
|     2|  89774|            MMA|2015-10-24 19:33:20|
+------+-------+---------------+-------------------+
only showing top 5 rows
```

Step 2:

I then used spark SQL to query the dataframes to get the required outputs. This involved creating tempviews of movies,ratings and tags

Q.1 Show the aggregated number of ratings per year

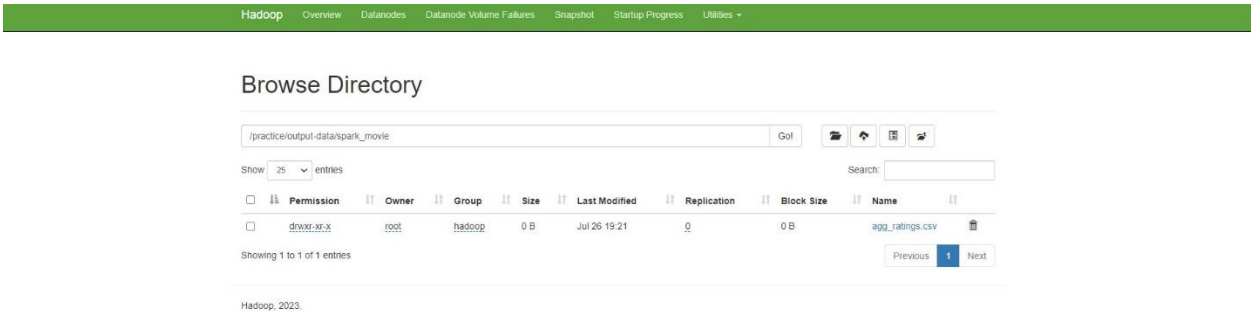
```
[16]: # Work with spark SQL

df_movies.createOrReplaceTempView("Movies")
df_ratings.createOrReplaceTempView("Ratings")
df_tags.createOrReplaceTempView("Tags")
```

```
•[19]: # Aggregated number of ratings per year

query = """ select year(timestamp) as year ,count(rating) as ratings from Ratings group by year(timestamp) order by year(timestamp) desc"""

output = spark.sql(query)
output.show()

# write data in HDFS into single file

output.coalesce(1).write.mode('overwrite').format('csv').option('header','true').option('delimiter',',').save('/practice/output-data/spark_movie/agg_ratings.csv')
print("write successfully")
```

```
write successfully
```

Step 3:

I made sure to save the output in a HDFS location. I also checked the files in the HDFS Namenode using the UI.



Q.2 Show the average monthly number of ratings

```
[20]: # avg number of ratings per month

query = """ select month(timestamp) as month ,avg(rating) as ratings from Ratings group by month(timestamp) order by month(timestamp) desc"""

output = spark.sql(query)
output.show()

# write data in HDFS into single file

output.coalesce(1).write.mode('overwrite').format('csv').option('header','true').option('delimiter',',').save('/practice/output-data/avg_ratings.csv')
print("write successfully")
```

```
+-----+------------------+
|month|           ratings|
+-----+------------------+
|   12|3.5149035651665694|
|   11| 3.634921455146755|
|   10|3.5126608841634024|
|    9|3.5998237367802584|
|    8|3.3590478289618693|
|    7| 3.639928057553957|
|    6| 3.4172237960339944|
|    5| 3.450565101534503|
|    4| 3.613498123463181|
|    3| 3.455518018018018|
|    2|3.3519973804846104|
|    1|  3.50374251497006|
+-----+------------------+

write successfully
```

Q.3 Show the rating levels distribution

[13]:
```
# Rating Level Distribution

query = """ with t1 as ( select
   rating,case when rating between 0 and 2 then '0.0 -2.0'
   when rating between 2.3 and 4 then '2.0-4.0'
   else '>4' end as rating_bucket from Ratings),

   t2 as (select rating_bucket,count(*) as counts
   from t1
   group by rating_bucket
   order by rating_bucket)

   select rating_bucket,counts,counts*100/sum(counts) over() as percentage from t2"""

output = spark.sql(query)
output.show(5)
```

```
24/07/26 16:56:06 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradation.
24/07/26 16:56:06 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradation.
24/07/26 16:56:06 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradation.
24/07/26 16:56:07 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradation.
24/07/26 16:56:07 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradation.
24/07/26 16:56:08 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradation.
24/07/26 16:56:08 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradation.
24/07/26 16:56:08 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradation.
24/07/26 16:56:08 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradation.
[Stage 41:>                                                         (0 + 1) / 1]
+-------------+------+------------------+
|rating_bucket|counts|        percentage|
+-------------+------+------------------+
|     0.0 -2.0| 13523|13.410885001388394|
|      2.0-4.0| 65551| 65.00753699075727|
|          >4| 21762|21.581578007854336|
+-------------+------+------------------+
```

Q.4 Show the 18 movies that are tagged but not rated

[17]:
```
# Movies Tagged but not Rated

query = """with cte as (
   select distinct t.movieID from Tags as t left join Ratings as r on t.movieID = r.movieID  where r.movieID  is null)

   select m.title  from Movies as m inner join cte as t on m.movieID = t.movieID order by 1"""

output = spark.sql(query)
output.show()

output.coalesce(1).write.mode('overwrite').format('csv').option('header','true').option('delimiter',',').save('/practice/output-data/movies_tagged.csv')
```

```
+--------------------+
|               title|
+--------------------+
|Browning Version,...|
|Call Northside 77...|
|  Chalet Girl (2011)|
|  Chosen, The (1981)|
|Color of Paradise...|
|For All Mankind (...|
|I Know Where I'm ...|
|In the Realms of ...|
|Innocents, The (1...|
|Mutiny on the Bou...|
|      Niagara (1953)|
|Parallax View, Th...|
|        Proof (1991)|
|Road Home, The (W...|
|Roaring Twenties,...|
|      Scrooge (1970)|
|This Gun for Hire...|
|Twentieth Century...|
+--------------------+
```

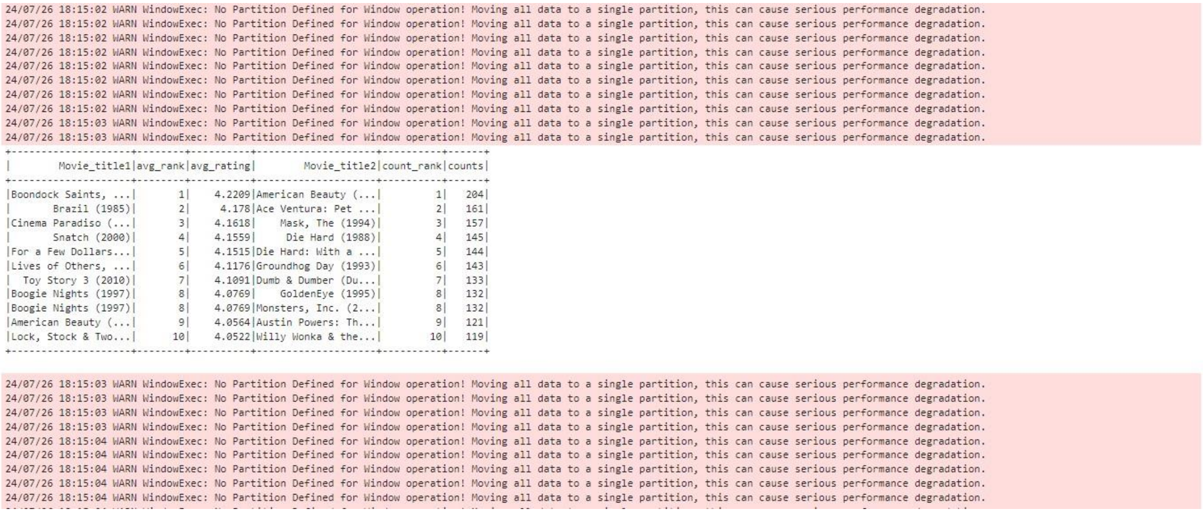
Q.5 Show the movies that have rating but no tag

```
[19]: # Movies Rated but not Tagged

query = """with cte as (
   select distinct r.movieID from Tags as t right join Ratings as r on t.movieID = r.movieID  where t.movieID  is null)

   select m.title  from Movies as m inner join cte as t on m.movieID = t.movieID order by 1"""

output = spark.sql(query)
output.show()

output.coalesce(1).write.mode('overwrite').format('csv').option('header','true').option('delimiter',',').save('/practice/output-data/movies_rating.csv')
print("write successfully")
```

```
+--------------------+
|               title|
+--------------------+
|          '71 (2014)|
|'Hellboy': The Se...|
|'Round Midnight (...|
| 'Salem's Lot (2004)|
|'Til There Was Yo...|
|'Tis the Season f...|
|  'burbs, The (1989)|
|'night Mother (1986)|
|*batteries not in...|
|...All the Marble...|
|00 Schneider - Ja...|
|   1-900 (06) (1994)|
|           10 (1979)|
|10 Cent Pistol (2...|
|10 Items or Less ...|
|     10 Years (2011)|
|    10,000 BC (2008)|
|    100 Girls (2000)|
|  100 Streets (2016)|
|101 Dalmatians II...|
+--------------------+
only showing top 20 rows

write successfully
```

Q.6 Fousing 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

```
[22]: # Rated but untagged movies (With more than 30 user ratings) -- Top Movies in terms of avg rating and number of ratings

query = """with t1 as (
      select movieid from ratings group by 1 having count(distinct userid)>30),

       t2 as (Select
           t1.movieID from t1
           left join TAGS as t
           on t1.movieID=t.movieID
           where t.movieID IS NULL),

        t3 as (Select m.title,m.movieID
           from MOVIES as m
          inner join t2
           on m.movieID=t2.movieID
           order by 1),

        t4 as (Select t3.title,avg(r.rating) as avg_rating,
           dense_rank()over(order by avg(r.rating) desc) as avg_rank
           from t3 left join RATINGS as r
           on t3.movieID=r.movieID
           group by 1),


            t5 as (Select t3.title,count(r.rating) as counts,
           dense_rank()over(order by count(r.rating) desc) as count_rank
           from t3 left join RATINGS as r
           on t3.movieID=r.movieID
           group by 1)

         Select t4.title as Movie_title1,t4.avg_rank,Round(t4.avg_rating,4) as avg_rating,t5.title as Movie_title2,t5.count_rank,t5.counts
           from t4 inner join t5
           on t4.avg_rank=t5.count_rank
           where t4.avg_rank<=10 and t5.count_rank<=10"""


output = spark.sql(query)
output.show()

# Write data in HDFS into single file

output.coalesce(1).write.mode("overwrite").format('csv').option('header', 'true') .option('delimiter', ',').save('/practice/top_10_avgratings&count_ratings.csv')
print("Write Successfull")
```



```
24/07/26 18:15:02 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradation.
24/07/26 18:15:02 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradation.
24/07/26 18:15:02 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradation.
24/07/26 18:15:02 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradation.
24/07/26 18:15:02 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradation.
24/07/26 18:15:02 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradation.
24/07/26 18:15:02 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradation.
24/07/26 18:15:02 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradation.
24/07/26 18:15:03 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradation.
24/07/26 18:15:03 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradation.
+--------------------+--------+----------+--------------------+----------+------+
|        Movie_title1|avg_rank|avg_rating|        Movie_title2|count_rank|counts|
+--------------------+--------+----------+--------------------+----------+------+
|Boondock Saints, ...|       1|    4.2209|American Beauty (...|         1|   204|
|       Brazil (1985)|       2|     4.178|Ace Ventura: Pet ...|         2|   161|
|Cinema Paradiso (...|       3|    4.1618|     Mask, The (1994)|         3|   157|
|       Snatch (2000)|       4|    4.1559|      Die Hard (1988)|         4|   145|
|For a Few Dollars...|       5|    4.1515|Die Hard: With a ...|         5|   144|
|Lives of Others, ...|       6|    4.1176|Groundhog Day (1993)|         6|   143|
|  Toy Story 3 (2010)|       7|    4.1091|Dumb & Dumber (Du...|         7|   133|
|Boogie Nights (1997)|       8|    4.0769|     GoldenEye (1995)|         8|   132|
|Boogie Nights (1997)|       8|    4.0769|Monsters, Inc. (2...|         8|   132|
|American Beauty (...|       9|    4.0564|Austin Powers: Th...|         9|   121|
|Lock, Stock & Two...|      10|    4.0522|Willy Wonka & the...|        10|   119|
+--------------------+--------+----------+--------------------+----------+------+

24/07/26 18:15:03 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradation.
24/07/26 18:15:03 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradation.
24/07/26 18:15:03 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradation.
24/07/26 18:15:03 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradation.
24/07/26 18:15:04 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradation.
24/07/26 18:15:04 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradation.
24/07/26 18:15:04 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradation.
24/07/26 18:15:04 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradation.
24/07/26 18:15:04 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradation.
```

Q.7 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?

```
[26]: # Tags per movie vs Tags per User

      query = """ with t1 as (
          select  '1' as key,round((sum(case when tag is not null then 1 else 0 end)/count(distinct movieid)),2) as tags_per_movies from Tags),

           t2 as (
          select '1' as key , round((sum(case when tag is not null then 1 else 0 end)/count(distinct userid)),2) as tags_per_user from Tags)

            Select t1.tags_per_movies,t2.tags_per_user,
                CASE WHEN tags_per_user>tags_per_movies THEN 'tags_per_user is higher'
                ELSE 'tags_per_movie is higher' END as Comparison
                from t1 inner join t2 on t1.key=t2.key"""

      output = spark.sql(query)
      output.show()

      # Write data in HDFS into single file

      output.coalesce(1).write.mode("overwrite").format('csv').option('header', 'true') .option('delimiter', ',').save('/practice/tags_per_movieVStags_per_user.csv')
      print("Write Successfull")

      +---------------+-------------+--------------------+
      |tags_per_movies|tags_per_user|          Comparison|
      +---------------+-------------+--------------------+
      |           2.34|         63.5|tags_per_user is ...|
      +---------------+-------------+--------------------+

      Write Successfull
```

Q.8 Identify the users that tagged movies without rating them

```
[30]: # Users that tagged but did not Rate movies

query= """

        Select distinct t.userid
        from TAGS as t
        left join RATINGS as r
        on t.movieID=r.movieID
        where r.userID is NULL"""


output = spark.sql(query)
output.show()

# Write data in HDFS into single file

output.coalesce(1).write.mode("overwrite").format('csv').option('header', 'true') .option('delimiter', ',').save('/practice/output-data/users_tagged_not_rate.csv')
print("Write Successfull")
```

```
+------+
|userid|
+------+
|   474|
|   288|
|   543|
|   318|
+------+

Write Successfull
```

Q.9 What is the average number of ratings per user in ratings DF? And the average number of ratings per movie?

```
[31]: # Ratings per user versus Ratings per Movie

query= """with t1 as(
          Select '1' as key, round((SUM(CASE when rating IS NOT NULL THEN 1 ELSE 0 END)/count(distinct userid)),2) as ratings_per_user
          from RATINGS),

          t2 as ( Select '1' as key, round((sum(CASE WHEN rating IS NOT NULL THEN 1 ELSE 0 END)/count(distinct movieid)),2) as ratings_per_movie
          from RATINGS)

          Select t1.ratings_per_user,t2.ratings_per_movie
          from t1 inner join t2 on t1.key=t2.key"""


output = spark.sql(query)
output.show()

# Write data in HDFS into single file

output.coalesce(1).write.mode("overwrite").format('csv').option('header', 'true') .option('delimiter', ',').save('/practice/output-data/ratings_per_userVSratings_per_movie.csv')
print("Write Successfull")
```

```
+----------------+-----------------+
|ratings_per_user|ratings_per_movie|
+----------------+-----------------+
|           165.3|            10.37|
+----------------+-----------------+

Write Successfull
```

Q.10 What is the predominant (frequency based) genre per rating level?

```
[17]: # Predominant Genre per rating level

query = """with t1 as (
 select r.rating,m.genres,count(*) as counts,
 dense_rank() over(partition by r.rating  order by count(*) desc) as ranker
 from Movies as m right join Ratings as r on m.movieID  = r.movieID group by 1,2)


 select rating,genres as most_frequently_genres from t1 where ranker = 1  order by rating desc """

output = spark.sql(query)
output.show()

# Write data in HDFS into single file

output.coalesce(1).write.mode("overwrite").format('csv').option('header', 'true') .option('delimiter', ',').save('/practice/output-data/freq_tag_per_genre.csv')
print("Write Successfull")
```

```
+------+-----------------------+
|rating|most_frequently_genres|
+------+-----------------------+
|   5.0|                  Drama|
|   4.5|                  Drama|
|   4.0|                  Drama|
|   3.5|                 Comedy|
|   3.0|                 Comedy|
|   2.5|                 Comedy|
|   2.0|                 Comedy|
|   1.5|                 Comedy|
|   1.0|                 Comedy|
|   0.5|                 Comedy|
+------+-----------------------+

Write Successfull
```

Q.11 What is the predominant tag per genre and the most tagged genres?

```
[18]: # Predominant tag per genre

query = """with t1 as(
select t.tag,m.genres ,count(*) as counts,
dense_rank() over (partition by m.genres order by count(*) desc) as ranker
from Movies as m left join Tags  as t on t.movieID = m.movieID group by 1,2 )

select genres,tag as most_frequent_tag from t1 where ranker =1  order by genres desc """

output = spark.sql(query)
output.show()


output.coalesce(1).write.mode("overwrite").format('csv').option('header', 'true') .option('delimiter', ',').save('/practice/output-data/freq_genre_per_rating.csv')
print("Write Successfull")
```

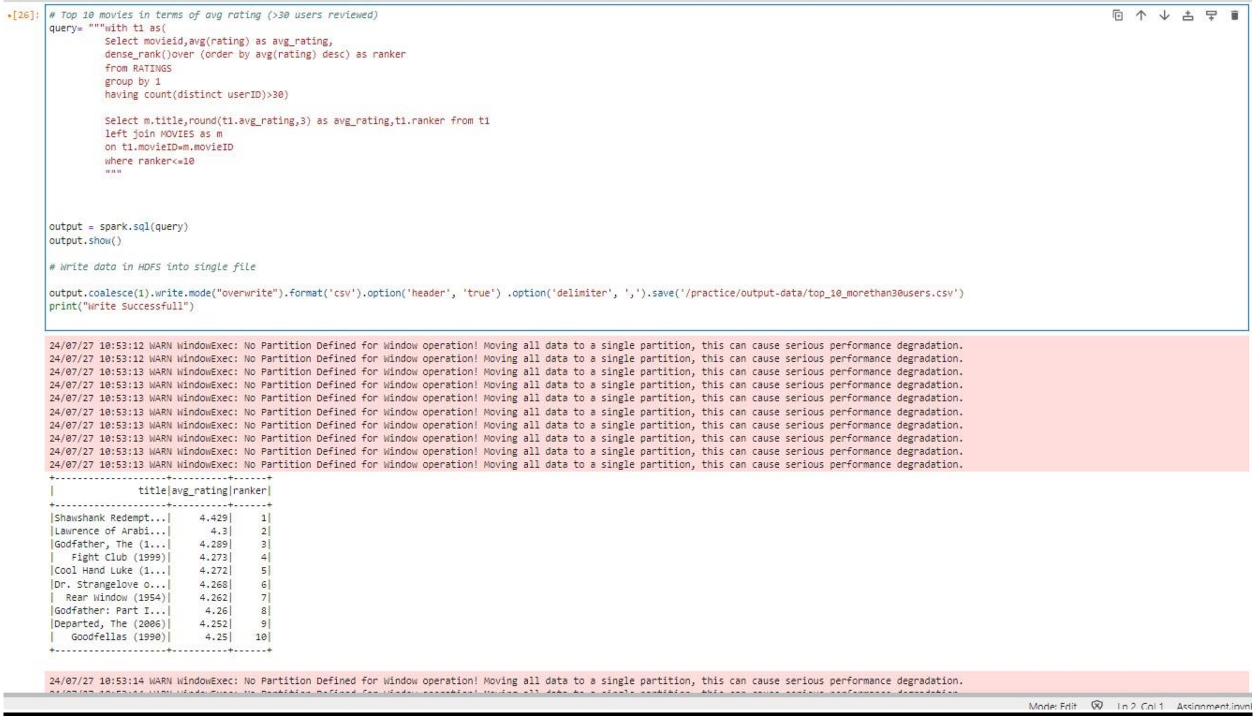
```
+--------------------+-----------------+
|              genres|most_frequent_tag|
+--------------------+-----------------+
|             Western|             NULL|
|                 War|             NULL|
|            Thriller|             NULL|
|Sci-Fi|Thriller|IMAX|             NULL|
|     Sci-Fi|Thriller|             NULL|
|         Sci-Fi|IMAX|           sci-fi|
|         Sci-Fi|IMAX|      time-travel|
|              Sci-Fi|             NULL|
|     Romance|Western|             NULL|
|         Romance|War|        Hemingway|
|    Romance|Thriller|             NULL|
|Romance|Sci-Fi|Th...|        artistic|
|Romance|Sci-Fi|Th...|             NULL|
|Romance|Sci-Fi|Th...|        Beautiful|
|Romance|Sci-Fi|Th...|      atmospheric|
|Romance|Sci-Fi|Th...|   existentialism|
|Romance|Sci-Fi|Th...|            artsy|
|Romance|Sci-Fi|Th...|        dreamlike|
|      Romance|Sci-Fi|             NULL|
|             Romance|             NULL|
+--------------------+-----------------+
only showing top 20 rows

Write Successfull
```

Q.12 What are the most predominant (popularity based) movies?

```
[23]: # Top 10 popular movies (most users seen/rated it)

query = """with t1 as (
select r.movieID,m.title,count(distinct r.userID) as counts,
dense_rank() over(order by count(r.userID) DESC ) AS ranker
from Ratings as r left join Movies as m on r.movieID = m.movieID group by 1,2 )

select title,counts from t1 where ranker<=10"""

output = spark.sql(query)
output.show()

# Write data in HDFS into single file

output.coalesce(1).write.mode("overwrite").format('csv').option('header', 'true') .option('delimiter', ',').save('/practice/output-data/popular_movies.csv')
print("Write Successfull")
```

```
24/07/27 10:50:08 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradation.
24/07/27 10:50:08 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradation.
24/07/27 10:50:08 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradation.
24/07/27 10:50:08 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradation.
24/07/27 10:50:08 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradation.
24/07/27 10:50:09 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradation.
24/07/27 10:50:09 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradation.
24/07/27 10:50:10 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradation.
24/07/27 10:50:10 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradation.
24/07/27 10:50:10 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradation.
24/07/27 10:50:10 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradation.

+--------------------+------+
|               title|counts|
+--------------------+------+
| Forrest Gump (1994)|   329|
|Shawshank Redempt...|   317|
| Pulp Fiction (1994)|   307|
|Silence of the La...|   279|
|  Matrix, The (1999)|   278|
|Star Wars: Episod...|   251|
|Jurassic Park (1993)|   238|
|   Braveheart (1995)|   237|
|Terminator 2: Jud...|   224|
|Schindler's List ...|   220|
+--------------------+------+
```

Q.13 Top 10 movies in terms of average rating (provided more than 30 users reviewed them)



```
# Top 10 movies in terms of avg rating (>30 users reviewed)
query= """with t1 as(
          Select movieid,avg(rating) as avg_rating,
          dense_rank()over (order by avg(rating) desc) as ranker
          from RATINGS
          group by 1
          having count(distinct userID)>30)

          Select m.title,round(t1.avg_rating,3) as avg_rating,t1.ranker from t1
          left join MOVIES as m
          on t1.movieID=m.movieID
          where ranker<=10
          """



output = spark.sql(query)
output.show()

# Write data in HDFS into single file

output.coalesce(1).write.mode("overwrite").format('csv').option('header', 'true') .option('delimiter', ',').save('/practice/output-data/top_10_morethan30users.csv')
print("Write Successfull")
```

```
24/07/27 10:53:12 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradation.
24/07/27 10:53:12 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradation.
24/07/27 10:53:13 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradation.
24/07/27 10:53:13 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradation.
24/07/27 10:53:13 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradation.
24/07/27 10:53:13 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradation.
24/07/27 10:53:13 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradation.
24/07/27 10:53:13 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradation.
24/07/27 10:53:13 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradation.
24/07/27 10:53:13 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradation.
+--------------------+----------+------+
|               title|avg_rating|ranker|
+--------------------+----------+------+
|Shawshank Redempt...|     4.429|     1|
|Lawrence of Arabi...|       4.3|     2|
|Godfather, The (1...|     4.289|     3|
|   Fight Club (1999)|     4.273|     4|
|Cool Hand Luke (1...|     4.272|     5|
|Dr. Strangelove o...|     4.268|     6|
|  Rear Window (1954)|     4.262|     7|
|Godfather: Part I...|      4.26|     8|
|Departed, The (2006)|     4.252|     9|
|   Goodfellas (1990)|      4.25|    10|
+--------------------+----------+------+

24/07/27 10:53:14 WARN WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradation.
```

Step 4 :

I made sure to check all the output of queries saved in a HDFS location. I also checked the files in the HDFS Namenode using the UI

Browse Directory

/practice/output-data Go!

Show 25 entries Search:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	root	hadoop	0 B	Jul 27 15:46	0	0 B	avg_ratings.csv
drwxr-xr-x	root	hadoop	0 B	Jul 27 16:06	0	0 B	freq_genre_per_rating.csv
drwxr-xr-x	root	hadoop	0 B	Jul 27 16:06	0	0 B	freq_tag_per_genre.csv
drwxr-xr-x	root	hadoop	0 B	Jul 26 23:19	0	0 B	movies_rating.csv
drwxr-xr-x	root	hadoop	0 B	Jul 26 23:14	0	0 B	movies_tagged.csv
drwxr-xr-x	root	hadoop	0 B	Jul 27 16:20	0	0 B	popular_movies.csv
drwxr-xr-x	root	hadoop	0 B	Jul 26 23:08	0	0 B	ratings_level.csv
drwxr-xr-x	root	hadoop	0 B	Jul 27 00:26	0	0 B	ratings_per_userVSratings_per_movie.csv
drwxr-xr-x	miralkunapara2003	hadoop	0 B	Jul 27 15:46	0	0 B	spark_movie
drwxr-xr-x	root	hadoop	0 B	Jul 27 00:23	0	0 B	tags_per_movieVStags_per_user.csv
drwxr-xr-x	root	hadoop	0 B	Jul 27 00:23	0	0 B	top_10_avgratings&count_ratings.csv
drwxr-xr-x	root	hadoop	0 B	Jul 27 16:23	0	0 B	top_10_morethan30users.csv
drwxr-xr-x	root	hadoop	0 B	Jul 27 00:23	0	0 B	users_tagged_not_rate.csv

Showing 1 to 13 of 13 entries

Previous 1 Next

Hadoop, 2023.