Name: Abhishek Obla Hema
Email: oh_abhishek@yahoo.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

[2]:
```
# Reading movies data

hdfs_path = '/tmp/spark_movie/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
```

[3]:
```
# Define the correct schema based on your CSV structure
schema = StructType([
    StructField("userId", IntegerType(), True),
    StructField("movieId", IntegerType(), True),
    StructField("rating", FloatType(), True),
    StructField("timestamp",IntegerType(), True),
])
hdfs_path = '/tmp/spark_movie/ratings.csv'
# Read the 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()
```

```
[Stage 3:>                                                          (0 + 1) / 1]
+------+-------+------+-------------------+
|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|
|     1|     70|   3.0|2000-07-30 18:40:00|
|     1|    101|   5.0|2000-07-30 18:14:28|
|     1|    110|   4.0|2000-07-30 18:36:16|
|     1|    151|   5.0|2000-07-30 19:07:21|
|     1|    157|   5.0|2000-07-30 19:08:20|
|     1|    163|   5.0|2000-07-30 19:00:50|
|     1|    216|   5.0|2000-07-30 18:20:08|
|     1|    223|   3.0|2000-07-30 18:16:25|
|     1|    231|   5.0|2000-07-30 18:19:39|
|     1|    235|   4.0|2000-07-30 18:15:08|
|     1|    260|   5.0|2000-07-30 18:28:00|
|     1|    296|   3.0|2000-07-30 18:49:27|
|     1|    316|   3.0|2000-07-30 18:38:30|
|     1|    333|   5.0|2000-07-30 18:19:39|
|     1|    349|   4.0|2000-07-30 18:42:43|
+------+-------+------+-------------------+
only showing top 20 rows
```

[4]:
```
# Define the correct schema based on your CSV structure
schema = StructType([
    StructField("userId", IntegerType(), True),
    StructField("movieId", IntegerType(), True),
    StructField("tag", StringType(), True),
    StructField("timestamp",IntegerType(), True),
])

hdfs_path = '/tmp/spark_movie/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()
```

```
+------+-------+----------------+-------------------+
|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|
|     2|  89774|       Tom Hardy|2015-10-24 19:33:25|
|     2| 106782|           drugs|2015-10-24 19:30:54|
|     2| 106782|Leonardo DiCaprio|2015-10-24 19:30:51|
|     2| 106782| Martin Scorsese|2015-10-24 19:30:56|
|     7|  48516|    way too long|2007-01-25 01:08:45|
|    18|    431|       Al Pacino|2016-05-01 21:39:25|
|    18|    431|        gangster|2016-05-01 21:39:09|
|    18|    431|           mafia|2016-05-01 21:39:15|
|    18|   1221|       Al Pacino|2016-04-26 19:35:06|
|    18|   1221|           Mafia|2016-04-26 19:35:03|
|    18|   5995|       holocaust|2016-02-17 18:57:52|
|    18|   5995|      true story|2016-02-17 18:57:59|
|    18|  44665|    twist ending|2016-03-02 19:51:23|
|    18|  52604| Anthony Hopkins|2016-03-10 22:58:16|
|    18|  52604| courtroom drama|2016-03-10 22:58:31|
+------+-------+----------------+-------------------+
only showing top 20 rows
```

Step 2:

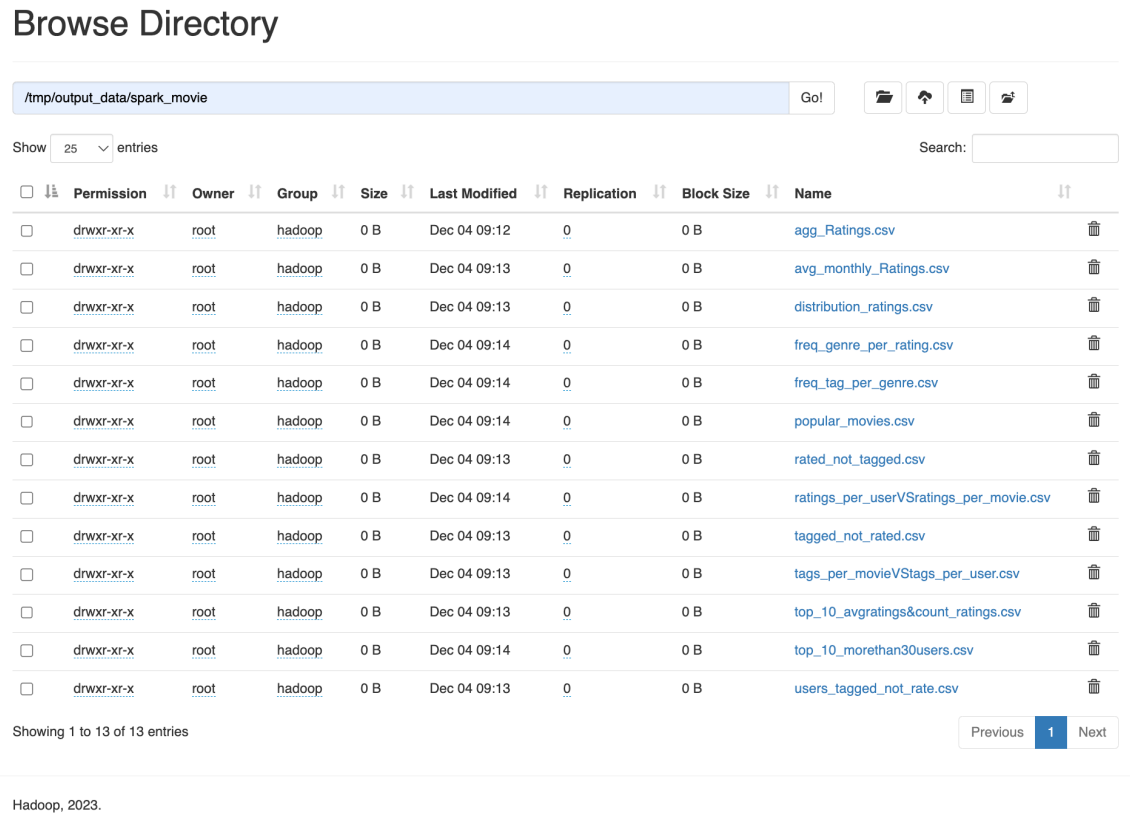
I then used spark SQL to query the dataframes to get the required outputs. This involved creating tempviews of movies,ratings and tags

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

df_movies.createOrReplaceTempView("MOVIES")
df_ratings.createOrReplaceTempView("RATINGS")
df_tags.createOrReplaceTempView("TAGS")
```

```
[6]: # Aggregated number of ratings per year

query= """Select year(timestamp) as year,count(rating) as ratings
       from RATINGS
       group by 1
        order by year(timestamp) desc"""


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

# Write data in HDFS into single file

# output.coalesce(1).write.format('csv').option('header', 'true').option('delimiter', ',').save('/tmp/output_data/spark_movie/')
output.coalesce(1).write.mode("overwrite").format('csv').option('header', 'true') .option('delimiter', ',').save('/tmp/output_data/spark_movie/agg_Ratings.c
print("Write Successfull")
```

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.



Browse Directory

/tmp/output_data/spark_movie Go!

Show 25 entries Search:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	root	hadoop	0 B	Dec 04 09:12	0	0 B	agg_Ratings.csv
drwxr-xr-x	root	hadoop	0 B	Dec 04 09:13	0	0 B	avg_monthly_Ratings.csv
drwxr-xr-x	root	hadoop	0 B	Dec 04 09:13	0	0 B	distribution_ratings.csv
drwxr-xr-x	root	hadoop	0 B	Dec 04 09:14	0	0 B	freq_genre_per_rating.csv
drwxr-xr-x	root	hadoop	0 B	Dec 04 09:14	0	0 B	freq_tag_per_genre.csv
drwxr-xr-x	root	hadoop	0 B	Dec 04 09:14	0	0 B	popular_movies.csv
drwxr-xr-x	root	hadoop	0 B	Dec 04 09:13	0	0 B	rated_not_tagged.csv
drwxr-xr-x	root	hadoop	0 B	Dec 04 09:14	0	0 B	ratings_per_userVSratings_per_movie.csv
drwxr-xr-x	root	hadoop	0 B	Dec 04 09:13	0	0 B	tagged_not_rated.csv
drwxr-xr-x	root	hadoop	0 B	Dec 04 09:13	0	0 B	tags_per_movieVStags_per_user.csv
drwxr-xr-x	root	hadoop	0 B	Dec 04 09:13	0	0 B	top_10_avgratings&count_ratings.csv
drwxr-xr-x	root	hadoop	0 B	Dec 04 09:14	0	0 B	top_10_morethan30users.csv
drwxr-xr-x	root	hadoop	0 B	Dec 04 09:13	0	0 B	users_tagged_not_rate.csv

Showing 1 to 13 of 13 entries Previous 1 Next

Hadoop, 2023.