Trần Duy Tân - 22550020

```
pip install findspark
```

```
Collecting findspark
  Downloading findspark-2.0.1-py2.py3-none-any.whl (4.4 kB)
Installing collected packages: findspark
Successfully installed findspark-2.0.1
```

```
pip install pyspark
```

```
Collecting pyspark
  Downloading pyspark-3.5.1.tar.gz (317.0 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 317.0/317.0 MB 3.0 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: py4j==0.10.9.7 in /usr/local/lib/python3.10/dist-packages (from pyspark) (0.10.9.7)
Building wheels for collected packages: pyspark
  Building wheel for pyspark (setup.py) ... done
  Created wheel for pyspark: filename=pyspark-3.5.1-py2.py3-none-any.whl size=317488491 sha256=f49c5d52cd3ca6bd82839478c66379e244e10417
  Stored in directory: /root/.cache/pip/wheels/80/1d/60/2c256ed38dddce2fdd93be545214a63e02fbd8d74fb0b7f3a6
Successfully built pyspark
Installing collected packages: pyspark
Successfully installed pyspark-3.5.1
```

câu 5 :Với mỗi userID, xuất ra (a) tên dòng phim nhận được đánh giá (rating) trung bình cao nhất từ user đó, (b) điểm rating trung bình của user đó cho dòng phim này, (c) danh sách top 5 bộ phim thuộc dòng phim này nhận được rating trung bình cao nhất từ mọi user mà user này chưa từng xem (chưa có đánh giá), và (c) điểm đánh giá trung bình của mọi user cho 5 bộ phim này. Ví dụ Film-Noir là dòng phim nhận được rating cao nhất từ user #1, bạn cần xuất ra danh sách 5 bộ phim thuộc về thể loại Film-Noir nhận rating trung bình cao nhất từ mọi user mà user #1 chưa xem (chưa đánh giá). Kết quả được sắp xếp theo userId như sau: (3 điểm)

Load Data

```
from pyspark.sql import SparkSession
from pyspark.sql import functions as F
from pyspark.sql.functions import avg, split, explode, col, collect_list, first
from pyspark.sql.types import StructType, StructField, StringType, IntegerType, DoubleType, ArrayType

spark = SparkSession.builder.appName("SparkDataframe").getOrCreate()

movieDF = spark.read.options(delimiter=',').schema('movieId INT, title STRING, genres STRING').csv("movies_small.csv")
ratingDF = spark.read.options(delimiter=',').schema('userId INT, movieId INT, rating DOUBLE, timestamp DOUBLE').csv(
    "ratings_small.csv")

join5 = ratingDF.join(movieDF, on=['movieId'], how="inner")
genresDF = join5.withColumn("genre", explode(split("genres", "\|")))
join_group5 = genresDF.groupBy("userId", "genre").agg(avg("rating").alias("avg_rating"))

max_avg_rating_per_user = join_group5.orderBy("avg_rating", ascending=False).groupBy("userId").agg(
    {"genre": "first", "avg_rating": "max"}).withColumnRenamed("max(avg_rating)", "User_avg_rating").withColumnRenamed(
    "first(genre)", "Highest_rated_genre_name").orderBy("userId")

max_avg_rating_per_user = max_avg_rating_per_user.withColumn("User_avg_rating", F.round("User_avg_rating", 2)).select("userId", "Highest_rate

genresDF = genresDF.withColumnRenamed("title", "genre_title")
genresJoinDF = movieDF.join(genresDF, on='movieId', how="inner")
avg_rating_per_movie = genresJoinDF.groupBy("movieId", "genre").agg(first("title").alias("title"), avg("rating").alias("avg_rating"))\
    .withColumn("avg_rating", F.round("avg_rating", 2))

schema = StructType([
    StructField("userId", IntegerType(), True),
    StructField("Top_5_unrated_movies_with_highest_rating", ArrayType(StringType()), True),
    StructField("Avg_rating_from_all_user", ArrayType(DoubleType()), True)
])

temp_df = spark.createDataFrame([], schema=schema)

# for x in max_avg_rating_per_user.rdd.collect():
for x in max_avg_rating_per_user.limit(100).rdd.collect():
    rated_movie_ids = ratingDF.select("movieId").where(col("userId") == x["userId"])
    notRated = ratingDF.select("movieId").subtract(rated_movie_ids)
    avg_rating_per_movie.join(notRated, "movieId", "inner")
```

```
    top5_movies = avg_rating_per_movie.select('movieId', 'title', 'avg_rating').where(col("genre") == x["Highest_rated_genre_name"]).orderBy(
    concatenated_titles_list = top5_movies.select(collect_list("title")).first()[0]
    concatenated_avg_rating_list = top5_movies.select(collect_list("avg_rating")).first()[0]
    temp_row = spark.createDataFrame([(x["userId"], concatenated_titles_list, concatenated_avg_rating_list)], ['Top_5_unrated_movies_with_hig
    temp_df = temp_df.union(temp_row)


# Hiển thị kết quả
result = max_avg_rating_per_user.join(temp_df, on=['userId'], how="inner").orderBy("userId")
result.show(10, truncate=False)
```

```
+------+------------------------+---------------+-----------------------------------------------------------------------------------------------
|userId|Highest_rated_genre_name|User_avg_rating|Top_5_unrated_movies_with_highest_rating
+------+------------------------+---------------+-----------------------------------------------------------------------------------------------
|1     |Film-Noir               |5.0            |[Rififi (Du rififi chez les hommes) (1955), Long Goodbye, The (1973), 13 Tzameti (2005
|2     |Romance                 |4.5            |[Man and a Woman, A (Un homme et une femme) (1966), Duel in the Sun (1946), Sandpiper,
|3     |Mystery                 |5.0            |[7 Faces of Dr. Lao (1964), Scooby-Doo! Curse of the Lake Monster (2010), Galaxy of Te
|4     |Horror                  |4.25           |[Slumber Party Massacre II (1987), Monster Squad, The (1987), Slumber Party Massacre I
|5     |Musical                 |4.4            |[Satin Rouge (2002), Woman Is a Woman, A (femme est une femme, Une) (1961), Babes in T
|6     |IMAX                    |4.67           |[More (1998), Happy Feet Two (2011), Journey to the West: Conquering the Demons (Daai
|7     |Horror                  |4.0            |[Slumber Party Massacre II (1987), Monster Squad, The (1987), Slumber Party Massacre I
|8     |Animation               |5.0            |[Ugly Duckling and Me!, The (2006), Idiots and Angels (2008), Lesson Faust (1994), Gar
|9     |Fantasy                 |5.0            |[Lesson Faust (1994), Light Years (Gandahar) (1988), Babes in Toyland (1934), Stand, T
|10    |Animation               |3.87           |[Ugly Duckling and Me!, The (2006), Idiots and Angels (2008), Lesson Faust (1994), Gar
+------+------------------------+---------------+-----------------------------------------------------------------------------------------------
only showing top 10 rows
```

LoadData

```
from pyspark.sql import SparkSession
from pyspark.sql import functions as f
spark = SparkSession.builder.appName("DataFrame").getOrCreate()
from pyspark.sql.functions import avg, explode, split
# Read movie and rating data
movieDF = spark.read.options(delimiter=',').schema('movie_id INT, title STRING, genres STRING').csv("movies_small.csv")
ratingDF = spark.read.options(delimiter=',').schema('user_id INT, movie_id INT, rating DOUBLE, timestamp STRING').csv("ratings_small.csv")
```

Câu 1 Xuất số lượng phim được làm mỗi năm. Kết quả được sắp xếp theo năm và có định dạng sau đây. (1 điểm)

```
temp1 = movieDF.withColumn("Year",
                           f.when(f.regexp_extract("title", r"\((\d{4}(\-\d{4})?)\)", 1) != "",
                                  f.regexp_extract("title", r"\((\d{4}(\-\d{4})?)\)", 1).cast("int"))
                            .otherwise(f.lit(' '))
                          )
temp1 = temp1.na.drop(subset=["Year"])
result1 = temp1.groupBy("Year").agg(f.count("year").alias("Num_of_movies"))
result1 = result1.sort(result1['Year'].cast('int'))
result1.show(20, truncate=False)
```

```
+----+-------------+
|Year|Num_of_movies|
+----+-------------+
|    |13           |
|1902|1            |
|1903|1            |
|1908|1            |
|1915|1            |
|1916|4            |
|1917|1            |
|1919|1            |
|1920|2            |
|1921|1            |
|1922|1            |
|1923|4            |
|1924|5            |
|1925|4            |
|1926|5            |
|1927|7            |
|1928|4            |
|1929|4            |
|1930|5            |
|1931|14           |
+----+-------------+
```

```
only showing top 20 rows
```

câu 2 : Xuất số lượng phim thuộc về mỗi dòng phim được làm mỗi năm. Kết quả được sắp xếp theo năm. Số phim của mỗi dòng được trình bày trong từng cột như sau: (2 điểm)

```
cau2 = movieDF.withColumn("Year", f.regexp_extract("title", r"\((\d{4})\)", 1))
cau2 = cau2.withColumn("Genre", f.explode(f.split("genres", "\|")))
result_cau2 = cau2.groupBy("Year").pivot("Genre").count()
result_cau2= result_cau2.orderBy("Year")
result_cau2.show(10, truncate=False)
```

```
+----+------------------+------+---------+---------+--------+------+-----+-----------+-----+-------+---------+------+----+-------+-----
|Year|(no genres listed)|Action|Adventure|Animation|Children|Comedy|Crime|Documentary|Drama|Fantasy|Film-Noir|Horror|IMAX|Musical|Myste
+----+------------------+------+---------+---------+--------+------+-----+-----------+-----+-------+---------+------+----+-------+-----
|    |9                 |1     |NULL     |NULL     |NULL    |NULL  |NULL |NULL       |2    |NULL   |NULL     |NULL  |NULL|NULL   |NULL
|1902|NULL              |1     |1        |NULL     |NULL    |NULL  |NULL |NULL       |NULL |1      |NULL     |NULL  |NULL|NULL   |NULL
|1903|NULL              |NULL  |NULL     |NULL     |NULL    |NULL  |1    |NULL       |NULL |NULL   |NULL     |NULL  |NULL|NULL   |NULL
|1908|NULL              |NULL  |NULL     |1        |NULL    |1     |NULL |NULL       |NULL |NULL   |NULL     |NULL  |NULL|NULL   |NULL
|1915|NULL              |NULL  |NULL     |NULL     |NULL    |NULL  |NULL |NULL       |1    |NULL   |NULL     |NULL  |NULL|NULL   |NULL
|1916|NULL              |1     |1        |NULL     |NULL    |1     |NULL |NULL       |1    |1      |NULL     |NULL  |NULL|NULL   |NULL
|1917|NULL              |NULL  |NULL     |NULL     |NULL    |1     |NULL |NULL       |NULL |NULL   |NULL     |NULL  |NULL|NULL   |NULL
|1919|NULL              |NULL  |NULL     |NULL     |NULL    |1     |NULL |NULL       |1    |NULL   |NULL     |NULL  |NULL|NULL   |NULL
|1920|NULL              |NULL  |NULL     |NULL     |NULL    |1     |1    |NULL       |NULL |1      |NULL     |1     |NULL|NULL   |NULL
|1921|NULL              |NULL  |NULL     |NULL     |NULL    |1     |NULL |NULL       |1    |NULL   |NULL     |NULL  |NULL|NULL   |NULL
+----+------------------+------+---------+---------+--------+------+-----+-----------+-----+-------+---------+------+----+-------+-----
only showing top 10 rows
```

Câu 3 Với mỗi userID, xuất điểm đánh giá trung bình của user đó cho mỗi dòng phim. Kết quả được sắp xếp theo userID. Kết quả của từng dòng phim được trình bày trong mỗi cột như sau: (2 điểm)

```
joinCau3 = ratingDF.join(movieDF, on=['movie_id'], how="inner")
cau3 = joinCau3.withColumn("Genre", explode(split("genres", "\|")))
result_cau3 = cau3.groupBy("user_id", "Genre").agg(avg("rating").alias("avg_rating"))
result_cau3 = result_cau3.groupBy("user_id").pivot("Genre").avg("avg_rating")
# Sắp xếp theo userId và hiển thị 100 dòng mà không truncate
sorted_cau3 = result_cau3.orderBy("user_id")
sorted_cau3.show(10, truncate=False)
```

```
+-------+------------------+------------------+------------------+------------------+------------------+------------------+------------
|user_id|(no genres listed)|Action            |Adventure         |Animation         |Children          |Comedy            |Crime
+-------+------------------+------------------+------------------+------------------+------------------+------------------+------------
|1      |NULL              |4.322222222222222 |4.3882352941176475|4.689655172413793 |4.5476190476190474|4.27710843373494  |4.3555555555
|2      |NULL              |3.9545454545454546|4.166666666666667 |NULL              |NULL              |4.0               |3.8
|3      |NULL              |3.5714285714285716|2.727272727272727 |0.5               |0.5               |1.0               |0.5
|4      |NULL              |3.32              |3.6551724137931036|4.0               |3.8               |3.5096153846153846|3.8148148148
|5      |NULL              |3.111111111111111 |3.25              |4.333333333333333 |4.111111111111111 |3.466666666666667 |3.8333333333
|6      |NULL              |3.609375          |3.893617021276596 |4.071428571428571 |3.617021276595745 |3.37007874015748  |3.2857142857
|7      |NULL              |3.2578125         |3.314814814814815 |3.392857142857143 |3.2               |3.163265306122449 |3.3076923076
|8      |NULL              |3.3333333333333335|3.5454545454545454|5.0               |4.25              |3.2083333333333335|3.8888888888
|9      |NULL              |3.125             |3.8               |4.0               |4.0               |3.6666666666666665|3.1428571428
|10     |NULL              |3.5               |3.5806451612903225|3.8666666666666667|3.607142857142857 |3.2658227848101267|3.1153846153
+-------+------------------+------------------+------------------+------------------+------------------+------------------+------------
only showing top 10 rows
```

Câu 4 :Với mỗi phim, xuất ra tên, năm, số lượng ratings, và điểm rating trung bình (từ mọi users) cho bộ phim đó. Kết quả được sắp xếp theo năm, sau đó theo tên phim như sau: (2 điểm)

```
cau4 = movieDF.join(ratingDF, on='movie_id', how='inner')
resultDF = cau4.groupBy('title', 'genres', 'movie_id').agg(
    f.count('rating').alias('Num_rating'),
    f.avg('rating').alias('Average_rating')
)
resultDF = resultDF.withColumn('Year', f.regexp_extract('title', r'\((\d{4})\)', 1))
resultDF = resultDF.select('Year', 'title', 'Num_rating', 'Average_rating').withColumnRenamed('title', 'Movie_name') \
    .orderBy(['Year', 'title'])
resultDF.show(10, truncate=False)
```

```
+----+--------------------------------+----------+--------------+
|Year|Movie_name                      |Num_rating|Average_rating|
```

```
+----+--------------------------------+----------+--------------+
|    |Babylon 5                       |2         |2.25          |
|    |Black Mirror                    |1         |5.0           |
|    |Cosmos                          |2         |4.5           |
|    |Death Note: Desu nôto (2006–2007)|1         |5.0           |
|    |Generation Iron 2               |1         |3.5           |
|    |Hyena Road                      |1         |2.0           |
|    |Maria Bamford: Old Baby         |1         |1.0           |
|    |Moonlight                       |1         |5.0           |
|    |Nocturnal Animals               |1         |3.0           |
|    |Paterson                        |1         |4.5           |
+----+--------------------------------+----------+--------------+
only showing top 10 rows
```