LAB 1 - Oleskevych Sofia, CS-314

Analysis part

I loaded “movies_metadata.csv” and ”ratings.csv” (or ratings_small.csv) as spark dataframes https://www.kaggle.com/rounakbanik/the-movies-dataset/data

Saved them to Google Drive, mounted GD to my colab notebook.

Using Python + Pyspark performed basic investigation and printed out summary (are there any missing values; How many records; How many unique users are there)

Preprocessing part

Using pyspark DataFrame API transform data loaded on a previous step to create MovieProfile and UserProfile entities with following schemas:

MovieProfile entity schema:

• MovieId - id of the movie

• MovieName - name of the movie

• MovieRating - average from all the user ratings for the movie

• Genres - a string with the comma separated genres for the movie

---

UserProfile entity schema:

• UserId - id of the user

• NumberOfMarks - how many marks user left

• YearsSpent - how many years has passed since the first mark

• AvgMark - average user mark

• AvgTimeBetweenMarks - how often user watches movies - how many time in average user spends between the marks in days (if I have 3 marks on the next dates: 21.07, 23.07 and 27.07, I will have (2 + 4) / 2 = 3 days in average)

• FavoriteGenre - favorite genre of a user (my own implementation)

```
!pip install pyspark
```

```
Collecting pyspark
  Downloading pyspark-3.2.0.tar.gz (281.3 MB)
     |████████████████████████████████| 281.3 MB 38 kB/s
Collecting py4j==0.10.9.2
  Downloading py4j-0.10.9.2-py2.py3-none-any.whl (198 kB)
```

```
     |████████████████████████████████| 198 kB 54.0 MB/s
Building wheels for collected packages: pyspark
  Building wheel for pyspark (setup.py) ... done
  Created wheel for pyspark: filename=pyspark-3.2.0-py2.py3-none-any.whl size=281805911
  Stored in directory: /root/.cache/pip/wheels/0b/de/d2/9be5d59d7331c6c2a7c1b6d1a4f463ce
Successfully built pyspark
Installing collected packages: py4j, pyspark
Successfully installed py4j-0.10.9.2 pyspark-3.2.0
```

```
from pyspark.sql import SparkSession

spark = SparkSession.builder.getOrCreate()


from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
raw_df = spark.read.option("header", "true").option("encoding", "UTF-8").csv("/content/drive/
raw_df2 = spark.read.option("header", "true").csv("/content/drive/MyDrive/ColabNotebooks/rati


raw_df.show()
```

```
+-----+--------------------+--------+--------------------+--------------------+-----+--
|adult|belongs_to_collection|  budget|              genres|            homepage|   id|
+-----+--------------------+--------+--------------------+--------------------+-----+--
|False| {'id': 10194, 'na...|30000000|[{'id': 16, 'name...|http://toystory.d...|  862|tt
|False|                null|65000000|[{'id': 12, 'name...|                null| 8844|tt
|False| {'id': 119050, 'n...|       0|[{'id': 10749, 'n...|                null|15602|tt
|False|                null|16000000|[{'id': 35, 'name...|                null|31357|tt
|False| {'id': 96871, 'na...|       0|[{'id': 35, 'name...|                null|11862|tt
|False|                null|60000000|[{'id': 28, 'name...|                null|  949|tt
|False|                null|58000000|[{'id': 35, 'name...|                null|11860|tt
|False|                null|       0|[{'id': 28, 'name...|                null|45325|tt
|False|                null|35000000|[{'id': 28, 'name...|                null| 9091|tt
|False| {'id': 645, 'name...|58000000|[{'id': 12, 'name...|http://www.mgm.co...|  710|tt
|False|                null|62000000|[{'id': 35, 'name...|                null| 9087|tt
|False|                null|       0|[{'id': 35, 'name...|                null|12110|tt
|False| {'id': 117693, 'n...|       0|[{'id': 10751, 'n...|                null|21032|tt
|False|                null|44000000|[{'id': 36, 'name...|                null|10858|tt
|False|                null|98000000|[{'id': 28, 'name...|                null| 1408|tt
|False|                null|52000000|[{'id': 18, 'name...|                null|  524|tt
|False|                null|16500000|[{'id': 18, 'name...|                null| 4584|tt
|False|                null| 4000000|[{'id': 80, 'name...|                null|    5|tt
|False| {'id': 3167, 'nam...|30000000|[{'id': 80, 'name...|                null| 9273|tt
|False|                null|60000000|[{'id': 28, 'name...|                null|11517|tt
+-----+--------------------+--------+--------------------+--------------------+-----+--
only showing top 20 rows
```

```
raw_df2.show()
```

```
+------+-------+------+----------+
|userId|movieId|rating| timestamp|
+------+-------+------+----------+
|     1|    110|   1.0|1425941529|
|     1|    147|   4.5|1425942435|
|     1|    858|   5.0|1425941523|
|     1|   1221|   5.0|1425941546|
|     1|   1246|   5.0|1425941556|
|     1|   1968|   4.0|1425942148|
|     1|   2762|   4.5|1425941300|
|     1|   2918|   5.0|1425941593|
|     1|   2959|   4.0|1425941601|
|     1|   4226|   4.0|1425942228|
|     1|   4878|   5.0|1425941434|
|     1|   5577|   5.0|1425941397|
|     1|  33794|   4.0|1425942005|
|     1|  54503|   3.5|1425941313|
|     1|  58559|   4.0|1425942007|
|     1|  59315|   5.0|1425941502|
|     1|  68358|   5.0|1425941464|
|     1|  69844|   5.0|1425942139|
|     1|  73017|   5.0|1425942699|
|     1|  81834|   5.0|1425942133|
+------+-------+------+----------+
only showing top 20 rows
```

```
# investigate schema in tree format
raw_df.printSchema()
```

```
root
 |-- adult: string (nullable = true)
 |-- belongs_to_collection: string (nullable = true)
 |-- budget: string (nullable = true)
 |-- genres: string (nullable = true)
 |-- homepage: string (nullable = true)
 |-- id: string (nullable = true)
 |-- imdb_id: string (nullable = true)
 |-- original_language: string (nullable = true)
 |-- original_title: string (nullable = true)
 |-- overview: string (nullable = true)
 |-- popularity: string (nullable = true)
 |-- poster_path: string (nullable = true)
 |-- production_companies: string (nullable = true)
 |-- production_countries: string (nullable = true)
 |-- release_date: string (nullable = true)
 |-- revenue: string (nullable = true)
 |-- runtime: string (nullable = true)
 |-- spoken_languages: string (nullable = true)
 |-- status: string (nullable = true)
 |-- tagline: string (nullable = true)
```

```
 |-- title: string (nullable = true)
 |-- video: string (nullable = true)
 |-- vote_average: string (nullable = true)
 |-- vote_count: string (nullable = true)
```

```
# count rows in dfs
raw_df.count(), raw_df2.count()
```

```
(45572, 26024289)
```

```
# check unique users

from pyspark.sql.functions import countDistinct, when, count, col, isnull, avg, aggregate

raw_df2.select(countDistinct("userId")).show()
```

```
+----------------------+
|count(DISTINCT userId)|
+----------------------+
|                270896|
+----------------------+
```

```
# check missing values in raw_df in each column

raw_df.select([count(when(isnull(c), c)).alias(c) for c in raw_df.columns]).show()
```

```
+-----+---------------------+------+------+--------+---+-------+-----------------+------
|adult|belongs_to_collection|budget|genres|homepage| id|imdb_id|original_language|origir
+-----+---------------------+------+------+--------+---+-------+-----------------+------
|    0|                40981|    17|    23|   37635| 31|    125|               45|
+-----+---------------------+------+------+--------+---+-------+-----------------+------
```

```
# rows of raw_df2 (ratings) where MovieId is determined

raw_df2.filter(raw_df2.movieId == 5).show()
```

```
+------+-------+------+----------+
|userId|movieId|rating| timestamp|
+------+-------+------+----------+
|     2|      5|   3.0| 867039249|
|    40|      5|   4.0| 862515493|
|    54|      5|   5.0| 986221889|
|    56|      5|   4.0|1410105373|
|    87|      5|   2.0| 867427886|
|    91|      5|   3.0| 838546731|
|    93|      5|   3.0| 835061814|
|   101|      5|   4.0| 850689925|
```

```
|   118|      5|   3.0| 859364983|
|   142|      5|   2.0| 866388486|
|   168|      5|   3.0| 850139631|
|   179|      5|   3.0| 863713756|
|   191|      5|   5.0| 854204045|
|   196|      5|   4.0| 849688746|
|   229|      5|   1.0|1037136141|
|   250|      5|   3.0| 995420603|
|   261|      5|   3.0| 864374769|
|   296|      5|   3.5|1111987460|
|   306|      5|   3.0| 846232643|
|   317|      5|   3.0| 855147687|
+------+-------+------+----------+
only showing top 20 rows
```

```
# 1 - MovieProfile creation

# cast the most important fields
mv_id_name = raw_df.select(raw_df.id.cast('int').alias('movieId'), raw_df.title.cast('string'


# basic preprocessing ( null values and valid movieName with regex)

def filtering(input_df):
    return input_df.filter(input_df.movieName.rlike('[a-zA-Z]+')) # one or more alpha symbols

def filtering2(input_df):
    return input_df.filter((input_df.movieName.startswith("[")==False)&(mv_id_name.movieId.is

df_m = mv_id_name.transform(filtering).transform(filtering2)
df_m.count()
```

```
42842
```

```
df_m.show()
```

```
+-------+--------------------+
|movieId|           movieName|
+-------+--------------------+
|    862|           Toy Story|
|   8844|             Jumanji|
|  15602|    Grumpier Old Men|
|  11862|Father of the Bri...|
|    949|                Heat|
|  11860|             Sabrina|
|  45325|        Tom and Huck|
|   9091|        Sudden Death|
|    710|           GoldenEye|
|   9087|The American Pres...|
|  12110|Dracula: Dead and...|
|  21032|               Balto|
|  10858|               Nixon|
```

```
|   1408|    Cutthroat Island|
|    524|              Casino|
|   4584|Sense and Sensibi...|
|      5|          Four Rooms|
|   9273|Ace Ventura: When...|
|  11517|         Money Train|
|   1710|             Copycat|
+-------+--------------------+
only showing top 20 rows
```

```
# select and cast from raw_df2 dataframe

mv_id_rate = raw_df2.select(raw_df2.movieId.cast('int').alias('movieId'), raw_df2.rating.cast


df_u = mv_id_rate.filter((mv_id_rate.movieId.isNotNull()) & (mv_id_rate.movieRating.isNotNull


# movie_id_name inner join movie_id_rating

mv_id_name_rate = df_m.join(df_u, df_m.movieId==df_u.movieId, 'inner')\
  .select(df_m.movieId.alias('movieId'), df_m.movieName.alias('movieName'), df_u.movieRating)
  .groupBy('movieId', 'movieName')\
  .agg(avg(df_u.movieRating).alias('movieRating'))



mv_id_name_rate.show()
```

```
+-------+--------------------+------------------+
|movieId|           movieName|       movieRating|
+-------+--------------------+------------------+
|     26|       Walk on Water| 3.614058355437666|
|   3469|Far from the Madd...| 4.065651760228354|
|   2029|              Tanguy| 3.463276836158192|
|    681|Diamonds Are Forever|3.5798319327731094|
|   5927|       Junior Bonner| 2.696526508226691|
|   8327|   The Holy Mountain|3.8284424379232505|
|   8094|The Magdalene Sis...| 3.815525876460768|
|  81782|         Kurukshetra|3.3834776334776335|
|  26792|      The Storm Gate|2.3130841121495327|
|   5602|      The Common Man|3.7735602094240837|
| 169864|      The Short Game| 3.247887323943662|
|   4441|               Candy| 3.259450171821306|
|   8653|            The Isle|2.5833333333333335|
|  32078|Panic in the Streets| 2.909448818897638|
|  26497|       Impy's Island|3.5555555555555554|
|     33|          Unforgiven|3.0555555555555554|
|   8744|    Albino Alligator|             3.125|
|  53883|             Anjathe| 3.992574257425743|
|   3073|Abbott and Costel...|3.3305555555555557|
|  70342|Beats Rhymes & Li...| 3.779220779220779|
+-------+--------------------+------------------+
```

```
only showing top 20 rows
```

```
movie_id_genres = raw_df.select(raw_df.id.cast('int').alias('movieId'), raw_df.genres.alias('

def filter_gen_id(input_df):
    return input_df.filter((input_df.movieId.isNotNull())&(input_df.genres.isNotNull()) )

movie_id_genres = movie_id_genres.transform(filter_gen_id)
movie_id_genres.count()
```

```
45363
```

```
import pyspark.sql.functions as f

final_df = movie_id_genres.select('movieId', f.get_json_object('genres', '$[*].name').alias('

final_df.show()
```

```
+-------+--------------------+
|movieId|              genres|
+-------+--------------------+
|    862|["Animation","Com...|
|   8844|["Adventure","Fan...|
|  15602|["Romance","Comedy"]|
|  31357|["Comedy","Drama"...|
|  11862|            "Comedy"|
|    949|["Action","Crime"...|
|  11860|["Comedy","Romance"]|
|  45325|["Action","Advent...|
|   9091|["Action","Advent...|
|    710|["Adventure","Act...|
|   9087|["Comedy","Drama"...|
|  12110| ["Comedy","Horror"]|
|  21032|["Family","Animat...|
|  10858| ["History","Drama"]|
|   1408|["Action","Advent...|
|    524|   ["Drama","Crime"]|
|   4584| ["Drama","Romance"]|
|      5|  ["Crime","Comedy"]|
|   9273|["Crime","Comedy"...|
|  11517|["Action","Comedy...|
+-------+--------------------+
only showing top 20 rows
```

```
from pyspark.sql.functions import col, concat_ws, split, regexp_replace

movie_fin  = final_df.withColumn("genres", regexp_replace(col("genres"), '[\\[\\]\\"]', ""))
```

```
movie_fin.show()
```

```
+-------+--------------------+
|movieId|              genres|
+-------+--------------------+
|    862|Animation,Comedy,...|
|   8844|Adventure,Fantasy...|
|  15602|      Romance,Comedy|
|  31357|Comedy,Drama,Romance|
|  11862|              Comedy|
|    949|Action,Crime,Dram...|
|  11860|      Comedy,Romance|
|  45325|Action,Adventure,...|
|   9091|Action,Adventure,...|
|    710|Adventure,Action,...|
|   9087|Comedy,Drama,Romance|
|  12110|       Comedy,Horror|
|  21032|Family,Animation,...|
|  10858|       History,Drama|
|   1408|    Action,Adventure|
|    524|         Drama,Crime|
|   4584|       Drama,Romance|
|      5|        Crime,Comedy|
|   9273|Crime,Comedy,Adve...|
|  11517| Action,Comedy,Crime|
+-------+--------------------+
only showing top 20 rows
```

```
movie_profile = movie_fin.join(mv_id_name_rate, movie_fin.movieId==mv_id_name_rate.movieId)\
  .select(mv_id_name_rate.movieId, mv_id_name_rate.movieName,\
          mv_id_name_rate.movieRating, movie_fin.genres)
```

```
# final preprocessed MovieProfile dataframe
```

```
movie_profile.show()
```

```
+-------+--------------------+------------------+--------------------+
|movieId|           movieName|       movieRating|              genres|
+-------+--------------------+------------------+--------------------+
|     26|       Walk on Water| 3.614058355437666|               Drama|
|   3469|Far from the Madd...| 4.065651760228354|       Drama,Romance|
|   2029|              Tanguy| 3.463276836158192|              Comedy|
|    681|Diamonds Are Forever|3.5798319327731094|Adventure,Action,...|
|   5927|       Junior Bonner| 2.696526508226691|Action,Drama,Western|
|   8327|   The Holy Mountain|3.8284424379232505|               Drama|
|   8094|The Magdalene Sis...| 3.815525876460768|               Drama|
|  81782|         Kurukshetra|3.3834776334776335|                null|
|  26792|      The Storm Gate|2.3130841121495327|       Drama,History|
|   5602|      The Common Man|3.7735602094240837|      Drama,Thriller|
| 169864|      The Short Game| 3.247887323943662|         Documentary|
|   4441|               Candy| 3.259450171821306|       Drama,Romance|
|   8653|            The Isle|2.5833333333333335|      Drama,Thriller|
|  32078|Panic in the Streets| 2.909448818897638|Drama,Thriller,Crime|
```

```
|  26497|       Impy's Island|3.5555555555555554|Animation,Family,...|
|     33|          Unforgiven|3.0555555555555554|             Western|
|   8744|    Albino Alligator|             3.125|Crime,Drama,Thriller|
|  53883|             Anjathe| 3.992574257425743|        Action,Drama|
|   3073|Abbott and Costel...|3.3305555555555557|       Comedy,Horror|
|  70342|Beats Rhymes & Li...| 3.779220779220779|   Music,Documentary|
+-------+--------------------+------------------+--------------------+
only showing top 20 rows
```

2 - UserProfile

```
from pyspark.sql.functions import col, min as min_, max as max_, from_utc_timestamp, from_uni

raw_df2 = raw_df2.withColumn('timestamp', from_unixtime(raw_df2.timestamp).alias("timestamp")

raw_df2.show()
```

```
+------+-------+------+-------------------+
|userId|movieId|rating|          timestamp|
+------+-------+------+-------------------+
|     1|    110|   1.0|2015-03-09 22:52:09|
|     1|    147|   4.5|2015-03-09 23:07:15|
|     1|    858|   5.0|2015-03-09 22:52:03|
|     1|   1221|   5.0|2015-03-09 22:52:26|
|     1|   1246|   5.0|2015-03-09 22:52:36|
|     1|   1968|   4.0|2015-03-09 23:02:28|
|     1|   2762|   4.5|2015-03-09 22:48:20|
|     1|   2918|   5.0|2015-03-09 22:53:13|
|     1|   2959|   4.0|2015-03-09 22:53:21|
|     1|   4226|   4.0|2015-03-09 23:03:48|
|     1|   4878|   5.0|2015-03-09 22:50:34|
|     1|   5577|   5.0|2015-03-09 22:49:57|
|     1|  33794|   4.0|2015-03-09 23:00:05|
|     1|  54503|   3.5|2015-03-09 22:48:33|
|     1|  58559|   4.0|2015-03-09 23:00:07|
|     1|  59315|   5.0|2015-03-09 22:51:42|
|     1|  68358|   5.0|2015-03-09 22:51:04|
|     1|  69844|   5.0|2015-03-09 23:02:19|
|     1|  73017|   5.0|2015-03-09 23:11:39|
|     1|  81834|   5.0|2015-03-09 23:02:13|
+------+-------+------+-------------------+
only showing top 20 rows
```

```
from pyspark.sql.functions import from_unixtime, to_timestamp

df_user_marks = raw_df2.select(raw_df2.userId.alias('userId'), raw_df2.rating, raw_df2.timest
                               .groupBy('userId')\
                               .agg(count(raw_df2.rating).alias('numberOfMarks'),\
                                    avg(raw_df2.rating).alias('avgMark'),\
                                    min_(raw_df2.timestamp).alias('firstMark'),\
```

```
                                    max_(raw_df2.timestamp).alias('lastMark') )
```

```
df_user_marks.count()
```

```
270896
```

```
df_user_marks.orderBy(df_user_marks.numberOfMarks.desc()).show()
```

```
+------+-------------+------------------+-------------------+-------------------+
|userId|numberOfMarks|           avgMark|          firstMark|           lastMark|
+------+-------------+------------------+-------------------+-------------------+
| 45811|        18276|3.1987579339023857|2015-12-15 06:59:27|2017-07-31 09:02:42|
|  8659|         9279|3.2784243991809463|2001-08-05 18:10:38|2015-12-24 20:44:49|
|270123|         7638|2.5974731605132235|2012-12-13 02:21:42|2017-08-04 03:03:56|
|179792|         7515|3.2083166999334662|2006-10-10 06:41:32|2015-02-12 06:35:57|
|228291|         7410|3.2201754385964914|2009-04-07 02:50:08|2017-08-04 00:28:21|
|243443|         6320|1.5760284810126581|2000-04-07 13:13:16|2017-08-01 16:26:43|
| 98415|         6094| 2.804972103708566|2002-07-05 04:57:01|2017-08-03 16:16:38|
|229879|         6024| 3.498256972111554|2001-09-10 15:15:01|2017-08-04 02:08:42|
| 98787|         5814|2.4380804953560373|2016-06-25 04:55:26|2017-03-30 21:30:34|
|172224|         5701|3.7478512541659357|2000-08-03 08:06:12|2016-12-13 06:03:58|
|230417|         5619|0.8632318917956932|2016-10-31 01:15:09|2017-08-02 02:42:08|
| 70648|         5356| 3.011669156086632|2006-11-13 17:54:17|2011-10-04 01:46:59|
|194690|         5206| 2.466192854398771|2006-01-09 07:59:42|2017-07-17 00:04:12|
|107720|         5169|2.4049139098471657|2002-08-19 19:48:36|2002-11-25 17:50:15|
| 24025|         4946|3.2052163364334816|2016-03-27 15:12:44|2016-03-27 16:10:44|
|165352|         4921|1.1382849014427963|2016-01-07 16:18:19|2017-03-27 07:13:06|
|243331|         4834|3.0393049234588334|1998-02-13 20:52:06|2017-08-02 02:43:20|
|101276|         4834| 2.863260239966901|2016-05-09 08:52:03|2016-05-09 09:36:29|
| 74275|         4815|2.9912772585669782|2012-09-27 01:52:35|2017-08-03 22:08:35|
| 41190|         4785|3.2189132706374086|2008-09-07 22:07:23|2012-01-08 03:12:52|
+------+-------------+------------------+-------------------+-------------------+
only showing top 20 rows
```

```
# count years from today to firstMark

from pyspark.sql import functions as f
from pyspark.sql import types as t

df_min_max = df_user_marks.withColumn('yearsSpent', f.datediff(f.current_date(), df_user_mark


# count avgTimeBetweenMarks in days

df_min_max_avg = df_min_max.withColumn('avgTimeBetweenMarks', f.datediff(df_min_max.lastMark,


# Find favourite genre:

"""
My approach:
```

```
1. Find string of all genres of movies watched by certain user
2. convert string splitted by comma to list
3. map list {Genre:Count}
4. select the most frequent Genre by Count
"""


# 1 step
from pyspark.sql.functions import sum as sum_

fav_df = raw_df2.join(movie_profile, raw_df2.movieId==movie_profile.movieId)\
  .select(raw_df2.userId, movie_profile.genres)\
  .groupBy(raw_df2.userId)\
  .agg(f.concat_ws(",", f.collect_list(movie_profile.genres)).alias('allGenres'))



fav_df.show()
```

```
+------+--------------------+
|userId|           allGenres|
+------+--------------------+
|   100|Drama,Action,Thri...|
|  1000|Comedy,Family,Mus...|
| 10000|TV Movie,Drama,Hi...|
|100000|Horror,Mystery,Ad...|
|100004|Crime,Drama,Myste...|
|100005|Drama,Thriller,Dr...|
|100006|Drama,Comedy,Dram...|
|100007|Drama,Thriller,Cr...|
|100008|Crime,Drama,Thril...|
|100009|Comedy,Thriller,C...|
| 10001|Drama,Mystery,Act...|
|100010|Drama,History,War...|
|100014|Drama,Action,Crim...|
|100015|Horror,Mystery,Ad...|
|100020|Drama,Drama,Drama...|
|100021|Drama,Thriller,Cr...|
|100022|Crime,Drama,Myste...|
|100025|Drama,Romance,Adv...|
|100029|Adventure,Drama,F...|
| 10003|Drama,Crime,Drama...|
+------+--------------------+
only showing top 20 rows
```

```
# 2 step

fav_df_list = fav_df.select(fav_df.userId, split(fav_df.allGenres, ",").alias("genresArray"))
fav_df_list.show()
```

```
+------+--------------------+
|userId|         genresArray|
+------+--------------------+
```

```
|   100|[Drama, Action, T...|
|  1000|[Comedy, Family, ...|
| 10000|[TV Movie, Drama,...|
|100000|[Horror, Mystery,...|
|100004|[Crime, Drama, My...|
|100005|[Drama, Thriller,...|
|100006|[Drama, Comedy, D...|
|100007|[Drama, Thriller,...|
|100008|[Crime, Drama, Th...|
|100009|[Comedy, Thriller...|
| 10001|[Drama, Mystery, ...|
|100010|[Drama, History, ...|
|100014|[Drama, Action, C...|
|100015|[Horror, Mystery,...|
|100020|[Drama, Drama, Dr...|
|100021|[Drama, Thriller,...|
|100022|[Crime, Drama, My...|
|100025|[Drama, Romance, ...|
|100029|[Adventure, Drama...|
| 10003|[Drama, Crime, Dr...|
+------+--------------------+
only showing top 20 rows
```

```
fav_df_list.count()
```

```
265049
```

```
# 3 step

temp = (fav_df_list.withColumn("Dist",f.array_distinct("genresArray"))
              .withColumn("Counts",f.expr("""transform(Dist,x->
                            aggregate(genresArray,0,(acc,y)-> IF (y=x, acc+1,acc))
                                      )""" ))
              .withColumn("Map",f.arrays_zip("Dist","Counts")
              ))
out = temp.withColumn("favoriteGenre",
                    f.expr("""element_at(array_sort(Map,(first,second)->
         CASE WHEN first['Counts']>second['Counts'] THEN -1 ELSE 1 END),1)['Dist']""" ))
out.show()
```

```
+------+--------------------+--------------------+--------------------+--------------------
|userId|         genresArray|                Dist|              Counts|
+------+--------------------+--------------------+--------------------+--------------------
|   100|[Drama, Action, T...|[Drama, Action, T...|     [2, 1, 1, 1, 1]|[{Drama, 2}, {Act
|  1000|[Comedy, Family, ...|[Comedy, Family, ...|  [1, 2, 1, 3, 1, 1]|[{Comedy, 1}, {Fa
| 10000|[TV Movie, Drama,...|[TV Movie, Drama,...|        [1, 2, 1, 1]|[{TV Movie, 1}, {
|100000|[Horror, Mystery,...|[Horror, Mystery,...|[2, 3, 5, 11, 1, ...|[{Horror, 2}, {My
|100004|[Crime, Drama, My...|[Crime, Drama, My...|[27, 81, 15, 33, ...|[{Crime, 27}, {Dr
|100005|[Drama, Thriller,...|[Drama, Thriller,...|        [3, 1, 1, 1]|[{Drama, 3}, {Thr
|100006|[Drama, Comedy, D...|[Drama, Comedy, R...|[6, 4, 1, 1, 1, 2...|[{Drama, 6}, {Com
|100007|[Drama, Thriller,...|[Drama, Thriller,...|[5, 4, 2, 3, 1, 1...|[{Drama, 5}, {Thr
|100008|[Crime, Drama, Th...|[Crime, Drama, Th...|[39, 99, 45, 5, 3...|[{Crime, 39}, {Dr
```

```
|100009|[Comedy, Thriller...|[Comedy, Thriller...|[4, 2, 1, 6, 2, 2...|[{Comedy, 4}, {Th
| 10001|[Drama, Mystery, ...|[Drama, Mystery, ...|[4, 2, 1, 1, 3, 1...|[{Drama, 4}, {Mys
|100010|[Drama, History, ...|[Drama, History, ...|[15, 2, 1, 4, 1, ...|[{Drama, 15}, {Hi
|100014|[Drama, Action, C...|[Drama, Action, C...|[2, 2, 2, 4, 2, 1...|[{Drama, 2}, {Act
|100015|[Horror, Mystery,...|[Horror, Mystery,...|[1, 2, 1, 5, 1, 2...|[{Horror, 1}, {My
|100020|[Drama, Drama, Dr...|[Drama, Thriller,...|[14, 4, 5, 6, 1, ...|[{Drama, 14}, {Th
|100021|[Drama, Thriller,...|[Drama, Thriller,...|[473, 226, 147, 7...|[{Drama, 473}, {T
|100022|[Crime, Drama, My...|[Crime, Drama, My...|[22, 75, 15, 30, ...|[{Crime, 22}, {Dr
|100025|[Drama, Romance, ...|[Drama, Romance, ...|[31, 12, 6, 2, 15...|[{Drama, 31}, {Ro
|100029|[Adventure, Drama...|[Adventure, Drama...|[2, 8, 2, 1, 2, 3...|[{Adventure, 2},
| 10003|[Drama, Crime, Dr...|[Drama, Crime, My...|[17, 8, 4, 5, 6, ...|[{Drama, 17}, {Cr
+------+--------------------+--------------------+--------------------+----------------
only showing top 20 rows
```

```
user_profile = df_min_max_avg.join(out, df_min_max_avg.userId==out.userId)\
  .select(df_min_max_avg.userId, df_min_max_avg.numberOfMarks,\
          df_min_max_avg.yearsSpent, df_min_max_avg.avgMark,\
          df_min_max_avg.avgTimeBetweenMarks, out.favoriteGenre )


# final UserProfile dataframe

user_profile.show()
```

```
+------+-------------+------------------+------------------+--------------------+-------
|userId|numberOfMarks|        yearsSpent|           avgMark| avgTimeBetweenMarks|  favor
+------+-------------+------------------+------------------+--------------------+-------
|   100|            3|12.714579055441478|3.3333333333333335|                 0.0|
|  1000|            6|21.284052019164957|               3.5|                 0.0|
| 10000|           10| 4.725530458590007|              4.25|                 0.0|
|100005|            5| 6.176591375770021|               4.4|                 0.0|
|100006|           17| 9.949349760438055| 3.676470588235294|                 0.0|
|100007|           18| 15.84394250513347|              3.25|                 0.0|
|100008|          307| 24.18069815195072| 3.996742671009772| 0.08496732026143791|
|100009|           21|13.905544147843942|3.3333333333333335|                 0.0|
| 10001|           11| 6.316221765913758|4.2727272727272725|                 0.0|
|100010|           36|24.908966461327857|3.8055555555555554|                 0.0|
|100014|           15|11.520876112251882|2.7666666666666666|                 0.0|
|100020|          100|  6.12457221081451|             3.425|0.010101010101010102|
|100021|         2216|16.520191649555098|3.1656137184115525|0.009932279909706547|
|100022|          292|19.802874743326488| 3.914383561643836|                 0.0|
|100025|          126| 21.56605065023956|3.5317460317460316|                 0.0|
|100029|           20|17.727583846680357|               3.5|                 0.0|
| 10003|           50|14.858316221765914|              2.53|                 0.0|
|100031|           53|   21.782340862423| 4.018867924528302|                 0.0|
|100039|           50|16.134154688569474|              3.42|                 0.0|
|100040|           16|13.708418891170432|               4.0|                 0.0|Science
+------+-------------+------------------+------------------+--------------------+-------
only showing top 20 rows
```

```
user_profile.count()
```

```
265049
```

```
# !pip install pyspark==<compatible-spark-version>
# !pyspark --packages io.delta:delta-core_2.12:1.1.0 --conf "spark.sql.extensions=io.delta.sq


# user_df.write.format("delta").mode("append").save("/tmp/delta/UserProfile")
# user_df.write.format("delta").mode("append").saveAsTable("default.UserProfile")
```