22520467_Lab2

March 19, 2025

[1]:
```
!pip install findspark
```

```
Collecting findspark
  Downloading findspark-2.0.1-py2.py3-none-any.whl.metadata (352 bytes)
Downloading findspark-2.0.1-py2.py3-none-any.whl (4.4 kB)
Installing collected packages: findspark
Successfully installed findspark-2.0.1
```

[2]:
```
mount_point = "/content/drive"
from google.colab import drive
drive.mount(mount_point)
```

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

[161]:
```
import findspark
findspark.init()
import pyspark
from pyspark.sql import SparkSession
from pyspark.sql.functions import from_unixtime
from datetime import datetime
```

[4]:
```
spark = SparkSession.builder.appName("Lab 2").getOrCreate()
```

[141]:
```
movies_rdd = spark.sparkContext.textFile("/content/drive/MyDrive/ds200/Lab2/
 ↪data/movies.txt")
occupation_rdd = spark.sparkContext.textFile("/content/drive/MyDrive/ds200/Lab2/
 ↪data/occupation.txt")
users_rdd = spark.sparkContext.textFile("/content/drive/MyDrive/ds200/Lab2/data/
 ↪users.txt")
ratings_1_rdd = spark.sparkContext.textFile("/content/drive/MyDrive/ds200/Lab2/
 ↪data/ratings_1.txt")
ratings_2_rdd = spark.sparkContext.textFile("/content/drive/MyDrive/ds200/Lab2/
 ↪data/ratings_2.txt")
```

[23]:
```
ratings_rdd = ratings_1_rdd.union(ratings_2_rdd)
```

0.0.1 Bài 1

[7]:
```
def parsed_movies_1(line):
  parts = line.split(",")
  movie_id = int(parts[0].strip())
  movie_title = parts[1].strip()
  return (movie_id, movie_title)
```

[8]:
```
def parsed_ratings_1(line):
  parts = line.split(",")
  movie_id = int(parts[1].strip())
  rating = float(parts[2].strip())
  return (movie_id, rating)
```

[174]:
```
movies_rdd_1 = movies_rdd.map(parsed_movies_1)
ratings_rdd_1 = ratings_rdd.map(parsed_ratings_1)
```

[183]:
```
joined_rdd_1 = movies_rdd_1.join(ratings_rdd_1)
```

[184]:
```
movie_ratings_rdd_1 = joined_rdd_1.map(lambda x: (x[1][0], (x[1][1], 1)))
ratings_summary_1 = movie_ratings_rdd_1.reduceByKey(lambda a, b: (a[0] + b[0],␣
 ↪a[1] + b[1]))
ratings_avg_1 = ratings_summary_1.mapValues(lambda x: (x[0] / x[1], x[1]))
```

[185]:
```
filtered_movies_1 = ratings_avg_1.filter(lambda x: x[1][1] >= 5)
highest_rated_movie_1 = filtered_movies_1.takeOrdered(1, key=lambda x: -x[1][0])
```

[186]:
```
for movie in ratings_avg_1.collect():
    movie_title, (avg_rating, total_ratings) = movie
    print(f"{movie_title} AverageRating: {avg_rating:.2f} (TotalRatings:␣
 ↪{total_ratings})")

print()

if highest_rated_movie_1:
    movie_title, (avg_rating, total_ratings) = highest_rated_movie_1[0]
    print(f"{movie_title} is the highest rated movie with an average rating of␣
 ↪{avg_rating:.2f} among movies with at least 5 ratings.")
else:
    print("No movie meets the criteria.")
```

```
Lawrence of Arabia (1962) AverageRating: 3.44 (TotalRatings: 18)
The Silence of the Lambs (1991) AverageRating: 3.14 (TotalRatings: 7)
Mad Max: Fury Road (2015) AverageRating: 3.47 (TotalRatings: 18)
The Godfather: Part II (1974) AverageRating: 4.00 (TotalRatings: 17)
The Lord of the Rings: The Return of the King (2003) AverageRating: 3.82
(TotalRatings: 11)
Fight Club (1999) AverageRating: 3.50 (TotalRatings: 7)
```

```
The Terminator (1984) AverageRating: 4.06 (TotalRatings: 18)
The Lord of the Rings: The Fellowship of the Ring (2001) AverageRating: 3.89
(TotalRatings: 18)
The Social Network (2010) AverageRating: 3.86 (TotalRatings: 7)
No Country for Old Men (2007) AverageRating: 3.89 (TotalRatings: 18)
E.T. the Extra-Terrestrial (1982) AverageRating: 3.67 (TotalRatings: 18)
Sunset Boulevard (1950) AverageRating: 4.36 (TotalRatings: 7)
Gladiator (2000) AverageRating: 3.61 (TotalRatings: 18)
Psycho (1960) AverageRating: 4.00 (TotalRatings: 2)

Sunset Boulevard (1950) is the highest rated movie with an average rating of
4.36 among movies with at least 5 ratings.
```

0.0.2 Bài 2

[72]:
```
def parsed_movies_2(line):
  parts = line.split(",")
  movie_id = int(parts[0].strip())
  movie_genre = parts[2].strip().split("|")
  return (movie_id, movie_genre)
```

[73]:
```
def parsed_ratings_2(line):
  parts = line.split(",")
  movie_id = int(parts[1].strip())
  rating = float(parts[2].strip())
  return (movie_id, rating)
```

[74]:
```
movies_rdd_2 = movies_rdd.map(parsed_movies_2)
ratings_rdd_2 = ratings_rdd.map(parsed_ratings_2)
```

[75]:
```
movies_rdd_2_flat = movies_rdd_2.flatMap(lambda x: [(x[0], genre) for genre in␣
 ↪x[1]])
joined_rdd_2 = movies_rdd_2_flat.join(ratings_rdd_2)
```

[76]:
```
genre_ratings_rdd_2 = joined_rdd_2.map(lambda x: (x[1][0], (x[1][1], 1)))
ratings_summary_2 = genre_ratings_rdd_2.reduceByKey(lambda a, b: (a[0] + b[0],␣
 ↪a[1] + b[1]))
ratings_avg_2 = ratings_summary_2.mapValues(lambda x: (x[0]/x[1], x[1]))
```

[77]:
```
for genre in ratings_avg_2.collect():
    movie_genre, (avg_rating, total_ratings) = genre
    print(f"{movie_genre} - AverageRating: {avg_rating:.2f} (TotalRatings:␣
 ↪{total_ratings})")
```

```
Sci-Fi - AverageRating: 3.73 (TotalRatings: 54)
Action - AverageRating: 3.71 (TotalRatings: 54)
Drama - AverageRating: 3.76 (TotalRatings: 128)
Family - AverageRating: 3.67 (TotalRatings: 18)
```

```
Biography - AverageRating: 3.56 (TotalRatings: 25)
Horror - AverageRating: 4.00 (TotalRatings: 2)
Fantasy - AverageRating: 3.86 (TotalRatings: 29)
Mystery - AverageRating: 4.00 (TotalRatings: 2)
Thriller - AverageRating: 3.70 (TotalRatings: 27)
Adventure - AverageRating: 3.63 (TotalRatings: 83)
Film-Noir - AverageRating: 4.36 (TotalRatings: 7)
Crime - AverageRating: 3.81 (TotalRatings: 42)
```

0.0.3 Bài 3

[115]:
```
def parsed_movies_3(line):
  parts = line.split(",")
  movie_id = int(parts[0].strip())
  movie_title = parts[1].strip()
  return (movie_id, movie_title)
```

[116]:
```
def parsed_ratings_3(line):
  parts = line.split(",")
  user_id = int(parts[0].strip())
  movie_id = int(parts[1].strip())
  rating = float(parts[2].strip())
  return (user_id, (movie_id, rating))
```

[117]:
```
def parsed_users_3(line):
  parts = line.split(",")
  user_id = int(parts[0].strip())
  user_gender = parts[1].strip()
  return (user_id, user_gender)
```

[118]:
```
movies_rdd_3 = movies_rdd.map(parsed_movies_3)
ratings_rdd_3 = ratings_rdd.map(parsed_ratings_3)
users_rdd_3 = users_rdd.map(parsed_users_3)
```

[119]:
```
users_ratings_3 = users_rdd_3.join(ratings_rdd_3)
users_ratings_3 = users_ratings_3.map(lambda x: (x[1][1][0],(x[1][0],␣
 ↪x[1][1][1])))
joined_rdd_3 = movies_rdd_3.join(users_ratings_3)
movie_ratings_rdd_3 = joined_rdd_3.map(lambda x: ((x[1][0], x[1][1][0]),␣
 ↪(x[1][1][1], 1)))
```

[120]:
```
ratings_summary_3 = movie_ratings_rdd_3.reduceByKey(lambda a, b: (a[0] + b[0],␣
 ↪a[1] + b[1]))
ratings_avg_3 = ratings_summary_3.map(lambda x: (x[0][0], (x[0][1], x[1][0]/
 ↪x[1][1])))
output_3 = ratings_avg_3.groupByKey().mapValues(lambda values: {"M": None, "F":␣
 ↪None, **dict(values)}).mapValues(lambda d: (d["M"], d["F"]))
```

[121]:
```
for temp in output_3.collect():
    movie_title, (male_avg_rating, female_avg_rating) = temp
    male_avg_str = f"{male_avg_rating:.2f}" if male_avg_rating is not None else␣
 ↪"NA"
    female_avg_str = f"{female_avg_rating:.2f}" if female_avg_rating is not␣
 ↪None else "NA"
    print(f"{movie_title} - Male_Avg: {male_avg_str}, Female_Avg:␣
 ↪{female_avg_str}")
```

```
The Social Network (2010) - Male_Avg: 4.00, Female_Avg: 3.67
Lawrence of Arabia (1962) - Male_Avg: 3.55, Female_Avg: 3.31
No Country for Old Men (2007) - Male_Avg: 3.92, Female_Avg: 3.83
The Silence of the Lambs (1991) - Male_Avg: 3.33, Female_Avg: 3.00
Gladiator (2000) - Male_Avg: 3.59, Female_Avg: 3.64
Mad Max: Fury Road (2015) - Male_Avg: 4.00, Female_Avg: 3.32
The Godfather: Part II (1974) - Male_Avg: 4.06, Female_Avg: 3.94
The Lord of the Rings: The Fellowship of the Ring (2001) - Male_Avg: 4.00,
Female_Avg: 3.80
The Lord of the Rings: The Return of the King (2003) - Male_Avg: 3.75,
Female_Avg: 3.90
Psycho (1960) - Male_Avg: NA, Female_Avg: 4.00
The Terminator (1984) - Male_Avg: 3.93, Female_Avg: 4.14
E.T. the Extra-Terrestrial (1982) - Male_Avg: 3.81, Female_Avg: 3.55
Fight Club (1999) - Male_Avg: 3.50, Female_Avg: 3.50
Sunset Boulevard (1950) - Male_Avg: 4.33, Female_Avg: 4.50
```

0.0.4 Bài 4

[122]:
```
def parsed_movies_4(line):
  parts = line.split(",")
  movie_id = int(parts[0].strip())
  movie_title = parts[1].strip()
  return (movie_id, movie_title)
```

[123]:
```
def parsed_ratings_4(line):
  parts = line.split(",")
  user_id = int(parts[0].strip())
  movie_id = int(parts[1].strip())
  rating = float(parts[2].strip())
  return (user_id, (movie_id, rating))
```

[124]:
```
def parsed_users_4(line):
  parts = line.split(",")
  user_id = int(parts[0].strip())
  user_age = parts[2].strip()
  return (user_id, user_age)
```

[125]:
```
def map_age_group(user_data):
    user_id, age = user_data
    age = int(age)
    if age <= 18:
        return (user_id, "0-18")
    elif age <= 35:
        return (user_id, "18-35")
    elif age <= 50:
        return (user_id, "35-50")
    else:
        return (user_id, "50+")
```

[134]:
```
movies_rdd_4 = movies_rdd.map(parsed_movies_4)
ratings_rdd_4 = ratings_rdd.map(parsed_ratings_4)
users_rdd_4 = users_rdd.map(parsed_users_4)
users_rdd_4_grouped = users_rdd_4.map(map_age_group)
```

[135]:
```
users_ratings_4 = users_rdd_4_grouped.join(ratings_rdd_3)
users_ratings_4 = users_ratings_4.map(lambda x: (x[1][1][0],(x[1][0],␣
 ↪x[1][1][1])))
joined_rdd_4 = movies_rdd_4.join(users_ratings_4)
movie_ratings_rdd_4 = joined_rdd_4.map(lambda x: ((x[1][0], x[1][1][0]),␣
 ↪(x[1][1][1], 1)))
```

[136]:
```
ratings_summary_4 = movie_ratings_rdd_4.reduceByKey(lambda a, b: (a[0] + b[0],␣
 ↪a[1] + b[1]))
ratings_avg_4 = ratings_summary_4.map(lambda x: (x[0][0], (x[0][1], x[1][0]/
 ↪x[1][1])))
output_4 = ratings_avg_4.groupByKey().mapValues(lambda values: {"0-18": None,␣
 ↪"18-35": None, "35-50": None, "50+": None, **dict(values)}).mapValues(lambda␣
 ↪d: (d["0-18"], d["18-35"], d["35-50"], d["50+"]))
```

[137]:
```
for temp in output_4.collect():
    movie_title, (age_0_18, age_18_35, age_35_50, age_50_plus) = temp
    age_0_18_str = f"{age_0_18:.2f}" if age_0_18 is not None else "NA"
    age_18_35_str = f"{age_18_35:.2f}" if age_18_35 is not None else "NA"
    age_35_50_str = f"{age_35_50:.2f}" if age_35_50 is not None else "NA"
    age_50_plus_str = f"{age_50_plus:.2f}" if age_50_plus is not None else "NA"
    print(f"{movie_title} - [0-18: {age_0_18_str}, 18-35: {age_18_35_str},␣
 ↪35-50: {age_35_50_str}, 50+: {age_50_plus_str}]")
```

```
The Social Network (2010) - [0-18: NA, 18-35: 4.00, 35-50: 3.67, 50+: NA]
Lawrence of Arabia (1962) - [0-18: NA, 18-35: 3.60, 35-50: 3.29, 50+: 4.50]
No Country for Old Men (2007) - [0-18: NA, 18-35: 3.81, 35-50: 3.94, 50+: 4.00]
The Silence of the Lambs (1991) - [0-18: NA, 18-35: 3.00, 35-50: 3.25, 50+: NA]
Gladiator (2000) - [0-18: NA, 18-35: 3.44, 35-50: 3.81, 50+: 3.50]
Mad Max: Fury Road (2015) - [0-18: NA, 18-35: 3.36, 35-50: 3.64, 50+: NA]
Psycho (1960) - [0-18: NA, 18-35: 4.50, 35-50: 3.50, 50+: NA]
```

```
The Godfather: Part II (1974) - [0-18: NA, 18-35: 3.78, 35-50: 4.25, 50+: NA]
The Lord of the Rings: The Fellowship of the Ring (2001) - [0-18: NA, 18-35:
4.00, 35-50: 3.83, 50+: NA]
The Lord of the Rings: The Return of the King (2003) - [0-18: NA, 18-35: 3.83,
35-50: 3.81, 50+: NA]
The Terminator (1984) - [0-18: NA, 18-35: 4.17, 35-50: 4.05, 50+: 3.75]
E.T. the Extra-Terrestrial (1982) - [0-18: NA, 18-35: 3.56, 35-50: 3.83, 50+:
3.00]
Fight Club (1999) - [0-18: NA, 18-35: 3.50, 35-50: 3.50, 50+: 3.50]
Sunset Boulevard (1950) - [0-18: NA, 18-35: 4.17, 35-50: 4.50, 50+: NA]
```

0.0.5 Bài 5

[154]:
```
def parsed_users_5(line):
  parts = line.split(",")
  user_id = int(parts[0].strip())
  occupation_id = int(parts[3].strip())
  return (occupation_id, user_id)
```

[155]:
```
def parsed_ratings_5(line):
  parts = line.split(",")
  user_id = int(parts[0].strip())
  rating = float(parts[2].strip())
  return (user_id, rating)
```

[156]:
```
def parsed_occupation_5(line):
  parts = line.split(",")
  occupation_id = int(parts[0].strip())
  occupation = parts[1].strip()
  return (occupation_id, occupation)
```

[157]:
```
occupation_rdd_5 = occupation_rdd.map(parsed_occupation_5)
ratings_rdd_5 = ratings_rdd.map(parsed_ratings_5)
users_rdd_5 = users_rdd.map(parsed_users_5)
```

[158]:
```
users_occupation_rdd_5 = users_rdd_5.join(occupation_rdd_5).map(lambda x:␣
 ↪(x[1][0], x[1][1]))
occupation_ratings_5 = users_occupation_rdd_5.join(ratings_rdd_5).map(lambda x:␣
 ↪(x[1][0], (x[1][1], 1)))
ratings_summary_5 = occupation_ratings_5.reduceByKey(lambda a, b: (a[0] + b[0],␣
 ↪a[1] + b[1]))
ratings_avg_5 = ratings_summary_5.mapValues(lambda x: (x[0] / x[1], x[1]))
```

[159]:
```
for temp in ratings_avg_5.collect():
    occupation, (avg_rating, total_ratings) = temp
    print(f"{occupation} - AverageRating: {avg_rating:.2f} (TotalRatings:␣
 ↪{total_ratings})")
```

```
Consultant - AverageRating: 3.86 (TotalRatings: 14)
Salesperson - AverageRating: 3.65 (TotalRatings: 17)
Engineer - AverageRating: 3.56 (TotalRatings: 18)
Manager - AverageRating: 3.47 (TotalRatings: 16)
Designer - AverageRating: 4.00 (TotalRatings: 13)
Doctor - AverageRating: 3.69 (TotalRatings: 21)
Artist - AverageRating: 3.73 (TotalRatings: 11)
Teacher - AverageRating: 3.70 (TotalRatings: 5)
Journalist - AverageRating: 3.85 (TotalRatings: 17)
Lawyer - AverageRating: 3.65 (TotalRatings: 17)
Nurse - AverageRating: 3.86 (TotalRatings: 11)
Student - AverageRating: 4.00 (TotalRatings: 8)
Accountant - AverageRating: 3.58 (TotalRatings: 6)
Programmer - AverageRating: 4.25 (TotalRatings: 10)
```

0.0.6 Bài 6

[162]:
```
def parsed_ratings_6(line):
  parts = line.split(",")
  timestamp = int(parts[3].strip())
  year = datetime.utcfromtimestamp(timestamp).year
  rating = float(parts[2].strip())
  return (year, (rating, 1))
```

[179]:
```
ratings_rdd_6 = ratings_rdd.map(parsed_ratings_6)
ratings_summary_6 = ratings_rdd_6.reduceByKey(lambda a, b: (a[0] + b[0], a[1] +␣
 ↪b[1]))
ratings_avg_6 = ratings_summary_6.mapValues(lambda x: (x[0] / x[1], x[1]))
```

[182]:
```
for year, (avg_rating, total_ratings) in ratings_avg_6.collect():
    print(f"{year} - TotalRatings: {total_ratings}, AverageRating: {avg_rating:.
 ↪2f}")
```

```
2020 - TotalRatings: 184, AverageRating: 3.75
```