Programming Data Science at Scale
Lab Session 4: Spark Data Processing

1 Introduction

This is the fourth lab session for the *Programming Data Science at Scale* course 2024/25. In this lab, you will use SPARK to process and analyze data from the Internet Movie Database (IMDB). You will implement three tasks using SPARK RDDs, based on the provided code template.

2 Data

For this lab, we provide you with a complete project template (`Lab4-template.zip`) which includes all required dependencies, configurations, and datasets. Your task is to implement the required functions in `src/main/scala/imdb/ImdbSpark.scala`.

2.1 Internet Movie Database (IMDB)

In this assignment, you will process a subset of the IMDB dataset using SPARK. The IMDB dataset provides extensive information about movies and TV shows. You will work with the following datasets:

- `title.basics.tsv`
- `title.ratings.tsv`
- `title.crew.tsv`
- `name.basics.tsv`

Please note that the datasets are in tab-separated values (TSV) format. Each line corresponds to a record, and fields are separated by tabs. Missing values are represented by `'\N'`. You should handle these missing values appropriately in your SPARK programs.

The schemas for the datasets are as follows:

3 Tasks

In this lab, you will implement the following three tasks using SPARK. You are provided with a code template in `ImdbSpark.scala`, where you will implement your solutions in the methods `task1`, `task2`, and

INDEX	FIELD	TYPE	DESCRIPTION
			title.basics.tsv
0	`tconst`	`String`	Unique title ID (e.g., `tt1234567`)
1	`titleType`	`String`	Type of title (e.g., `movie`, `short`)
2	`primaryTitle`	`String`	Primary title
3	`originalTitle`	`String`	Original title
4	`isAdult`	`Boolean`	`0` (non-adult) or `1` (adult)
5	`startYear`	`Int`	Release year
6	`endYear`	`Int`	End year (for TV series)
7	`runtimeMinutes`	`Int`	Runtime in minutes
8	`genres`	`String`	Genres (comma-separated)
			title.ratings.tsv
0	`tconst`	`String`	Unique title ID (e.g., `tt1234567`)
1	`averageRating`	`Double`	Average rating
2	`numVotes`	`Int`	Number of votes
			title.crew.tsv
0	`tconst`	`String`	Unique title ID
1	`directors`	`String`	Director(s) IDs (comma-separated)
2	`writers`	`String`	Writer(s) IDs (comma-separated)
			name.basics.tsv
0	`nconst`	`String`	Unique name ID (e.g., `nm1234567`)
1	`primaryName`	`String`	Person's primary name
2	`birthYear`	`Int`	Birth year
3	`deathYear`	`Int`	Death year
4	`primaryProfession`	`String`	Primary profession(s) (comma-separated)
5	`knownForTitles`	`String`	Known for titles (comma-separated)

Table 1: IMDB Dataset Schemas

`task3`. Use only simple SPARK RDD transformations and actions (maximum 2-3 collective APIs per task). You are **only allowed to use RDDs**; do not use DataFrames or Datasets.

Please make sure to handle missing values (`'\N'`) appropriately in your solutions.

Task 1

◁ **Task**

Calculate the average runtime of *movies* and *tvSeries* for each genre, and list the top 5 genres with the longest average runtime.

Return Type: `RDD[(String, Float)]`
Fields:

- `genre: String`
- `averageRunTime: Float`

Instructions:

1. Filter `titleBasicsRDD` to include only records where `titleType` is either `'movie'` or `'tvSeries'`.
2. Split the `genres` field into individual genres.
3. Map each record to pairs of genre and runtime.
4. Group by `genre` and calculate the average `runtimeMinutes`.
5. Order the results by `averageRunTime` in descending order and take the top 5.

Task 2

◁ **Task**

Task 2: For each decade from 2000 to 2020, find the *movies* with the highest ratings, considering only *movies* with at least 2 milion votes.

Return Type: `RDD[(Int, List[(String, Float)])]`
Fields:

- `decade: Int`
- `movies: List[(String, Float)]`, where each element is:
  - `movie: String`
  - `rating: Float`

Instructions:

1. Filter `titleBasicsRDD` to include only `'movie'` titles released between 2000 and 2020.
2. Filter `titleRatingsRDD` to include only movies with `numVotes` at least 2 milion votes.
3. Join the two RDDs on `tconst`.
4. Map to create a `(decade, (movieTitle, averageRating))` pair.
5. Group by `decade`.
6. For each decade, select the top movies based on `averageRating`.

Task 3

◁ **Task**

Task 3: List the top 10 writers with the highest average ratings for their films, considering only *movies* with a rating over 8.5 and at least 10,000 votes.

Return Type: `RDD[(String, Float)]`
Fields:

- `writerName: String`
- `averageRating: Float`

Instructions:

1. Filter `titleBasicsRDD` to include only `'movie'` titles.
2. Filter `titleRatingsRDD` to include only movies with `averageRating` over 8.5 and `numVotes` at least 10,000.
3. Join the filtered RDDs with `titleCrewRDD` on `tconst`.
4. Map to separate multiple writer IDs in the `writers` field.
5. Join with `nameBasicsRDD` on `nconst` to get writer names.
6. Group by `writerName` and calculate the average `averageRating`.
7. Filter to include writers with at least three movies.
8. Order by `averageRating` and take the top 10.