Big Data: Spark Exercise

Part 1: RDDs

First, import the relevant pyspark modules.

In [2]:

```
from pyspark import SparkConf, SparkContext
```

Next, create an instance of Spark context handler.

In [3]:

```
conf = SparkConf().setMaster("local").setAppName("SparkExample")
sc = SparkContext(conf = conf)
```

Word Count

1. Count the number of distinct words in `columns.csv` using pyspark.
2. Display the top 10 most frequent words

Hints:

- You need to read the file into an RDD object.
- **Recall**: an RDD file is a collection of lines.

- We don't want a collection of lines, we want a collection of words.
- **Recall**: The Python string method .split(), and the RDD' .flatMap() transformation, may be useful.

- You can use .countByValue() on the RDD to count the entries.
- You can call `sorted()` on a dict with a function to provide the value to sort on

- You want most frequent *words*, so normalize the input (ie., ignore case, ignore punctuation)

In [23]:

```
# read file
poem = sc.textFile('columns.csv')

# turn a collection of lines into a collection of words
words = poem.flatMap(lambda line : line.strip().replace('"', '').lower().split())

# count the number of words
freq = words.countByValue()

top = sorted(results, key=lambda k : freq[k], reverse=True)
```

In [26]:

```
top[:10]
```

Out[26]:

```
['i', 'to', 'of', 'and', 'my', 'the', 'in', 'always', 'a', 'am']
```

In [31]:

```
{freq[w]: w for w in top[:10]}
```

Out[31]:

```
{85: 'i',
 25: 'to',
 16: 'of',
 14: 'my',
 13: 'the',
 12: 'in',
 11: 'always',
 9: 'am'}
```

DataFrames

Movies

The "ml-100k" folder contains 100,000 movie ratings given by 943 users on 1682 movies.

The description of the data set is given in ml-100k/README.

The main dataset containing the ratings is "u.data", which is a space separated value (SSV) file with columns:

1. user id
2. movie id
3. rating
4. timestamp

The film titles are contained in "u.item".

Question

Find the top five most popular movies.

These are **not** necessarily the highest rated ones, but the movies most frequently rated by the users.

HINTS:

- groupBy().count() to get the number of ratings for each movie
- .orderBy() to sort the count column

In [26]:

```
from pyspark.sql import SparkSession
from pyspark.sql.types import StructField, StructType, StringType, ShortType, LongType
```

- Define `spark` using `SparkSession`

In [31]:

```
spark = (
SparkSession.builder
    .master("local")
    .appName("Movies")
    .getOrCreate()
)
```

- `spark.read` the `u.data` file

HINT:

- You will need to define your own schema:
  - "UserID" will be a `LongType`
  - "MoveID" will be a `LongType()`
  - "Rating" will be a `ShortType()`
  - "Time" will be a `LongType`
- Add the option `"delimiter"` , `"\t"`

In [29]:

```
schema_user = StructType([
  StructField("UserID", LongType()),
  StructField("MovieID", LongType()),
  StructField("Rating", ShortType()),
  StructField("Time", LongType()),
])

dfu = (
spark
    .read
    .schema(schema_user)
    .option("header", "false")
    .option("delimiter", "\t")
    .csv("ml-100k/u.data")
)

dfu.show(3)
```

```
+------+-------+------+---------+
|UserID|MovieID|Rating|     Time|
+------+-------+------+---------+
|   196|    242|     3|881250949|
|   186|    302|     3|891717742|
|    22|    377|     1|878887116|
+------+-------+------+---------+
only showing top 3 rows
```

- Read in `u.items`

HINT:

- You will need to define your own schema:
  - "MovieID" will be a `LongType`
  - "Title" will be a `StringType()`
- Add the option `"delimiter"` , `"|"`
- (You only need the first two columns for this problem, so that schema is complete)

In [36]:

```
schema_item = StructType([
  StructField("MovieID", LongType()),
  StructField("Title", StringType()),
])

dfi = (
spark
    .read
    .schema(schema_item)
    .option("header", "false")
    .option("delimiter", "|")
    .csv("ml-100k/u.item")
)

dfi.show(3)
```

```
+-------+-----------------+
|MovieID|            Title|
+-------+-----------------+
|      1| Toy Story (1995)|
|      2| GoldenEye (1995)|
|      3|Four Rooms (1995)|
+-------+-----------------+
only showing top 3 rows
```

- Group, Count and Order

In [47]:

```
results = dfu.groupby('MovieID').count().orderBy('count', ascending=0)
results.show(10)
```

```
+-------+-----+
|MovieID|count|
+-------+-----+
|     50|  583|
|    258|  509|
|    100|  508|
|    181|  507|
|    294|  485|
|    286|  481|
|    288|  478|
|      1|  452|
|    300|  431|
|    121|  429|
+-------+-----+
only showing top 10 rows
```

EXTRA:

- Join your results to your items data frame

HINT:

- The join condition is `results['MovieID'] == dfi['MovieID']`
- You can `.select` columns to tidy up the join

In [39]:

```
(results
    .join(dfi, results['MovieID'] == dfi['MovieID'])
    .select("Title", "count")
    .orderBy('count', ascending=0)
    .show()
)
```

```
+--------------------+-----+
|               Title|count|
+--------------------+-----+
|    Star Wars (1977)|  583|
|      Contact (1997)|  509|
|        Fargo (1996)|  508|
|Return of the Jed...|  507|
|    Liar Liar (1997)|  485|
|English Patient, ...|  481|
|       Scream (1996)|  478|
|    Toy Story (1995)|  452|
|Air Force One (1997)|  431|
|Independence Day ...|  429|
|Raiders of the Lo...|  420|
|Godfather, The (1...|  413|
| Pulp Fiction (1994)|  394|
|Twelve Monkeys (1...|  392|
|Silence of the La...|  390|
|Jerry Maguire (1996)|  384|
|    Rock, The (1996)|  378|
|Empire Strikes Ba...|  367|
|Star Trek: First ...|  365|
|Back to the Futur...|  350|
+--------------------+-----+
only showing top 20 rows
```

Extra: Find the best rated movies

Continuing from above, we now wish to find the top five highest rated films.

The best films are defined as those that have both the highest average rating **then** the most votes.

- `groupBy` MovieID
- use `.agg()` to explictly aggregate both the mean and count for Rating
- `orderBy` both the average and count (orderBy takes multiple columns and a list for `ascending`)
- `join` with the items on MovieID, and `select` , as above
- `filter` so each film has a rating count of at least, eg., 100

In [44]:

```
from pyspark.sql.functions import mean, count
```

In [51]:

```
results = (
    dfu.select('MovieID', 'Rating')
      .groupBy('MovieID')
      .agg(
        mean('Rating').alias('Er'),
        count('Rating').alias('Nr')
      )
)

(results
    .join(dfi, results['MovieID'] == dfi['MovieID'])
    .select("Title", "Er", "Nr")
    .filter(results["Nr"] > 100)
    .orderBy('Er', 'Nr', ascending=[0,0])
    .show(10)
)
```

```
+--------------------+------------------+---+
|               Title|                Er| Nr|
+--------------------+------------------+---+
|Close Shave, A (1...| 4.491071428571429|112|
|Schindler's List ...| 4.466442953020135|298|
|Wrong Trousers, T...| 4.466101694915254|118|
|   Casablanca (1942)|  4.45679012345679|243|
|Shawshank Redempt...| 4.445229681978798|283|
|  Rear Window (1954)|4.3875598086124405|209|
|Usual Suspects, T...| 4.385767790262173|267|
|    Star Wars (1977)|4.3584905660377355|583|
| 12 Angry Men (1957)|             4.344|125|
| Citizen Kane (1941)| 4.292929292929293|198|
+--------------------+------------------+---+
only showing top 10 rows
```