Jared Girouard – SEC01 (NUID 002981834)

Big Data System Engineering with Scala  
Fall 2022   
Assignment No. 7



**-List of Tasks Implemented**

In this assignment, we learned how to use Spark to process a csv into a DataFrame and then manipulate the DataFrame to calculate various values. The following things were accomplished in this assignment:

* The spark-csv module was modified with code being added to the MovieDatabaseAnalyzer.scala
* Tests were added to the MovieDatabaseAnalyzerTest.scala file for the class implemented in this assignment
* The MovieDatabaseAnalyzer object was modified to instantiate a MovieRatingAnalyzer class which creates a DataFrame with the movie database csv, select the “imdb\_rating” column, and calculate the mean and standard deviation of the IMDB rating for every movie.
* A method was added in the MovieDatabaseAnalyzer to take in a DataFrame and return a processed DataFrame containing the mean and stddev.

**-Code**

The setup was the most difficult part of this assignment for me. This was the first time I used Spark within IntelliJ, previously I only had used it in the command line. To start this assignment, I reused the spark-csv module which already existed within the CSYE7200 project. The build.sbt file in this module just needed some tweaking for the dependencies we would be working with. Specifically, we were working with Spark version 3.2.1 and Scala 2.12.12.

![Text

Description automatically generated]()

The main code of this project lived in the MovieDatabaseAnalyzer.scala. The first thing I did was test that Spark was working by creating a Spark session. I attempted to do a simple operation with my Spark session but was continuously getting an exception saying “Exception in thread "main" java.lang.IllegalAccessError: class org.apache.spark.storage.StorageUtils$ (in unnamed module @0x7bba5817) cannot access class sun.nio.ch.DirectBuffer (in module java.base) because module java.base does not export [sun.nio.ch](http://sun.nio.ch/) to unnamed module @0x7bba5817”. I looked up this exception and there were several solution where people added certain VM options to their run configurations. I tried all of these but was still receiving the error and unfortunately wasted many hours trying to figure it out. I reached out to the Professor, and he suggested downloading an older JDK like Java 8. After downloading Java 8, Spark started working, and I was able to get started on the assignment.

Since I was modifying an existing project, I did not want to remove too much of the existing functionality; however, I did remove the code from MovieDatabaseAnalyzer Object. My first step was to get the path for the movie\_metadata.csv file. This was simple one I referenced how it was done in the WordCount assignment code. I then created a case class MovieRatingAnalyzer which took in the path as a string. In this class, I created a new spark session and read in the CSV file using Spark via the method described in the assignment. When reading in the CSV, I used .option to tell Spark that the first row of the CSV was the header. I then passed the resulting DataFrame into a method that I created called scoreProcessor. This method felt unnecessary because the processing was only a single line of code; however, I wanted to follow the assignment which stated, “Create a method that accepts a DataFrame and returns the processed DataFrame.” This didn’t really make sense to me because all of the processing was a single line and did not feel like it needed to be abstracted into a method. The scoreProcessor function took in the movie DataFrame and ran 2 calculations. With Spark SQL I was able to do both calculations in one line. The first calculation was a mean calculation, where all I had to do was call “mean” and pass in the name of the column I wanted the mean from which was “imdb\_score”. The same was true for the standard deviation where I called stddev\_pop and passed in the column name. From researching online, I decided to use the population standard deviation because this was not just a sample of a larger dataset this was the entire population. I put both of these values into a DataFrame which then got returned to my class. I don’t think it was necessary for me to put them into a DataFrame, but I thought it was a good way to show them and got me more practice with DataFrames. I showed this DataFrame and got the numbers mean = 6.453200745804848 and stddev = 0.9984966998015917.

**![Text

Description automatically generated]()**

**-Unit tests**

This was the first assignment where we had to write our own unit tests. I kept the existing test that was in the MovieDatabaseAnaylzerTest.scala and just added two of my own. The first test checks that the MovieRatingAnalyzer is able to get the movie-metdata.csv and checks the resulting DataFrame. I tested that the count of all the rows is 1609 as we would expect and then show the first 10 rows. For my second test, I tested the DataFrame containing the mean and stddev from my MovieRatingAnalyzer class. I checked that the values within the DataFrame matched the expected values. This was a good opportunity to figure out how to access the actual values within a DataFrame.

**![Text

Description automatically generated]()**

**- Result**

This assignment was helpful as a simple introduction to working with Spark and manipulating DataFrames with SparkSQL. It was also the first time we had to write our own unit tests which was good experience.

Link to repository: <https://github.com/JaredGirouard/CSYE7200>