

Thejas Bharadwaj – SEC01 (NUID 002727189)

# Big Data System Engineering with Scala

## Spring 2023

### Assignment No. 7



## -List of Tasks Implemented

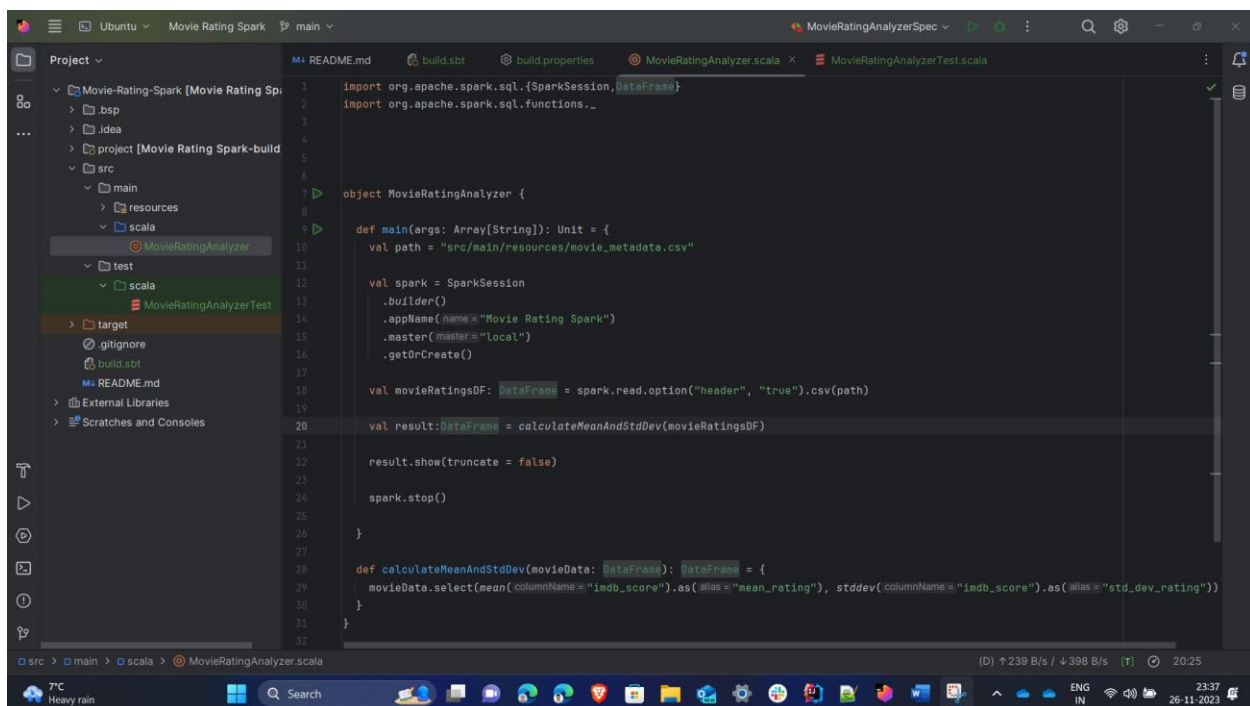
- 1) Created a Github repo - [thejas98/Movie-Rating-Spark \(github.com\)](https://github.com/thejas98/Movie-Rating-Spark)
- 2) Created two code files – main and test
- 3) The main file has the code to ingest the csv and a function which calculates the mean and the standard deviation of the 'imdb\_score' column.
- 4) The test file contains 2 test cases – One creates a sample df and tests if the function from main works correctly

## -Findings and analysis

N/A

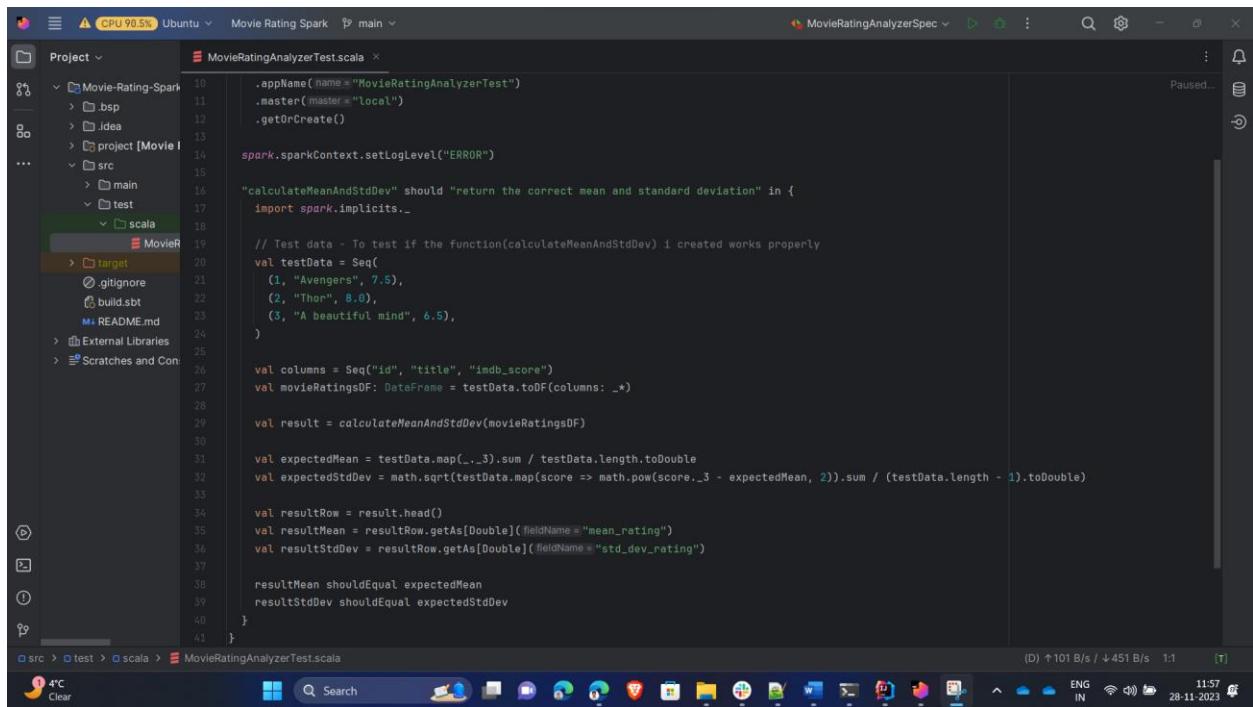
## -Code

### Main file



```
1 import org.apache.spark.sql.{SparkSession, DataFrame}
2 import org.apache.spark.sql.functions._
3
4
5
6
7 object MovieRatingAnalyzer {
8
9   def main(args: Array[String]): Unit = {
10     val path = "src/main/resources/movie_metadata.csv"
11
12     val spark = SparkSession
13       .builder()
14       .appName("Movie Rating Spark")
15       .master("local")
16       .getOrCreate()
17
18     val movieRatingsDF: DataFrame = spark.read.option("header", "true").csv(path)
19
20     val result: DataFrame = calculateMeanAndStdDev(movieRatingsDF)
21
22     result.show(truncate = false)
23
24     spark.stop()
25   }
26
27   def calculateMeanAndStdDev(movieData: DataFrame): DataFrame = {
28     movieData.select(mean(columnName = "imdb_score").as("mean_rating"), stddev(columnName = "imdb_score").as("std_dev_rating"))
29   }
30 }
31
32
```

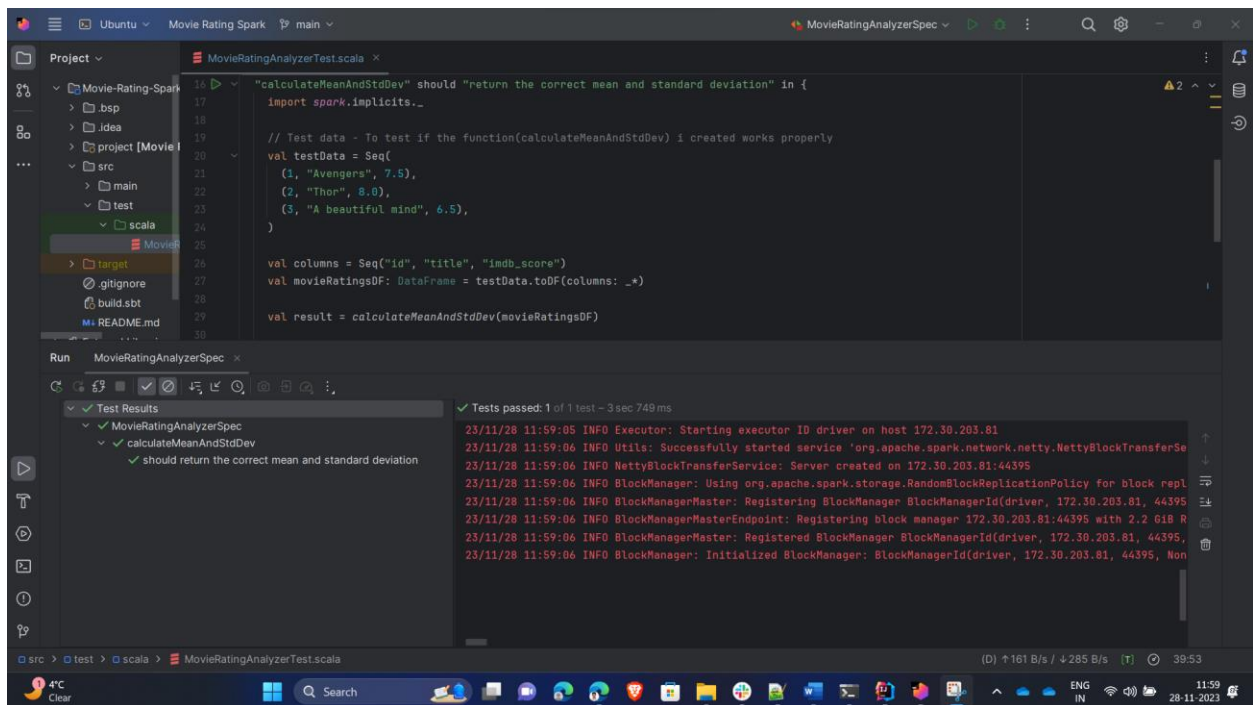
## Test file



The screenshot shows an IDE window titled "MovieRatingSpark" with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with folders like "src", "test", and "target". The code editor displays the file "MovieRatingAnalyzerTest.scala". The code defines a test class "MovieRatingAnalyzerTest" with a "test" method. The test method sets the Spark context log level to "ERROR", creates a test dataset with three rows of movie ratings, and then calls the "calculateMeanAndStdDev" function. The function returns a result row, and the test asserts that the mean and standard deviation values are correct.

```
10 .appName(name = "MovieRatingAnalyzerTest")
11 .master(master = "local")
12 .getOrCreate()
13
14 spark.sparkContext.setLogLevel("ERROR")
15
16 "calculateMeanAndStdDev" should "return the correct mean and standard deviation" in {
17   import spark.implicits._
18
19   // Test data - To test if the function(calculateMeanAndStdDev) I created works properly
20   val testData = Seq(
21     (1, "Avengers", 7.5),
22     (2, "Thor", 8.0),
23     (3, "A beautiful mind", 6.5),
24   )
25
26   val columns = Seq("id", "title", "imdb_score")
27   val movieRatingsDF: DataFrame = testData.toDF(columns: _*)
28
29   val result = calculateMeanAndStdDev(movieRatingsDF)
30
31   val expectedMean = testData.map(_._3).sum / testData.length.toDouble
32   val expectedStdDev = math.sqrt(testData.map(score => math.pow(score._3 - expectedMean, 2)).sum / (testData.length - 1)).toDouble
33
34   val resultRow = result.head()
35   val resultMean = resultRow.getAs[Double](fieldName = "mean_rating")
36   val resultStdDev = resultRow.getAs[Double](fieldName = "std_dev_rating")
37
38   resultMean shouldEqual expectedMean
39   resultStdDev shouldEqual expectedStdDev
40 }
41 }
```

## -Unit tests



The screenshot shows the same IDE window as before, but with the "Run" button clicked. The output console at the bottom shows the test results. The test "calculateMeanAndStdDev" passed, and the output shows the Spark context log level set to "ERROR". The test results are displayed in a table with columns for the test name, status, and duration.

```
Run MovieRatingAnalyzerSpec
Test Results
MovieRatingAnalyzerSpec
  calculateMeanAndStdDev
    ✓ should return the correct mean and standard deviation
Tests passed: 1 of 1 test - 3 sec 749 ms
23/11/28 11:59:05 INFO Executor: Starting executor ID driver on host 172.30.203.81
23/11/28 11:59:06 INFO Utils: Successfully started service 'org.apache.spark.network.netty.NettyBlockTransferService' on host 172.30.203.81
23/11/28 11:59:06 INFO NettyBlockTransferService: Server created on 172.30.203.81:44395
23/11/28 11:59:06 INFO BlockManager: Using org.apache.spark.storage.RandomBlockReplicationPolicy for block replication policy
23/11/28 11:59:06 INFO BlockManagerMaster: Registering BlockManager BlockManagerId(driver, 172.30.203.81, 44395)
23/11/28 11:59:06 INFO BlockManagerMasterEndpoint: Registering block manager 172.30.203.81:44395 with 2.2 GiB RAM
23/11/28 11:59:06 INFO BlockManagerMaster: Registered BlockManager BlockManagerId(driver, 172.30.203.81, 44395)
23/11/28 11:59:06 INFO BlockManager: Initialized BlockManager: BlockManagerId(driver, 172.30.203.81, 44395, Non
```