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)

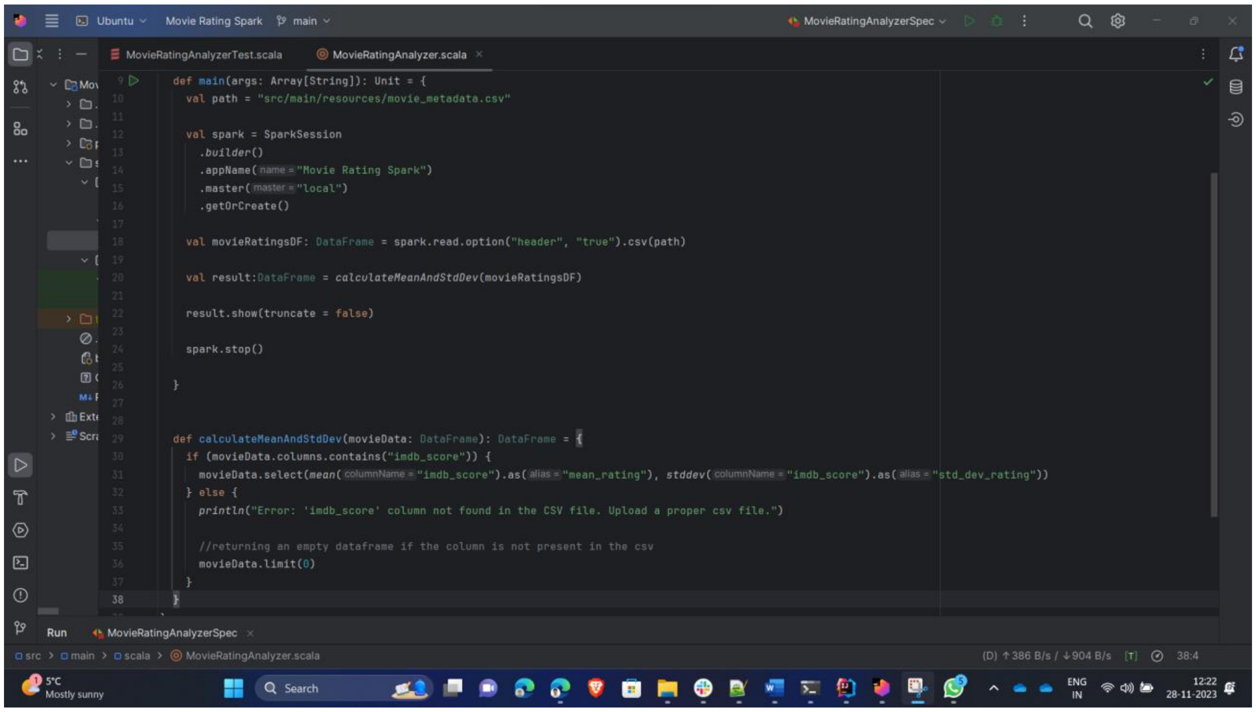
2) Created two code files – main and test

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 calculatemean from main works correctly and the other makes sure if you upload a csv without a ‘imdb_score’ column, it handles the error gracefully.

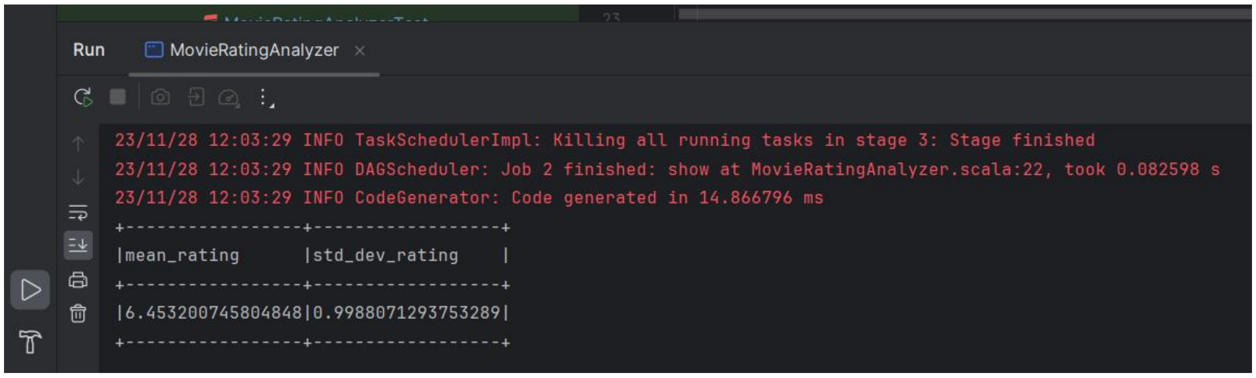
-Code

1) Main file



```scala
def main(args: Array[String]): Unit = {
  val path = "src/main/resources/movie_metadata.csv"

  val spark = SparkSession
    .builder()
    .appName("Movie Rating Spark")
    .master("local")
    .getOrCreate()

  val movieRatingsDF: DataFrame = spark.read.option("header", "true").csv(path)

  val result:DataFrame = calculateMeanAndStdDev(movieRatingsDF)

  result.show(truncate = false)

  spark.stop()

}


def calculateMeanAndStdDev(movieData: DataFrame): DataFrame = {
  if (movieData.columns.contains("imdb_score")) {
    movieData.select(mean("imdb_score").as("mean_rating"), stddev("imdb_score").as("std_dev_rating"))
  } else {
    println("Error: 'imdb_score' column not found in the CSV file. Upload a proper csv file.")

    //returning an empty dataframe if the column is not present in the csv
    movieData.limit(0)
  }
}
```

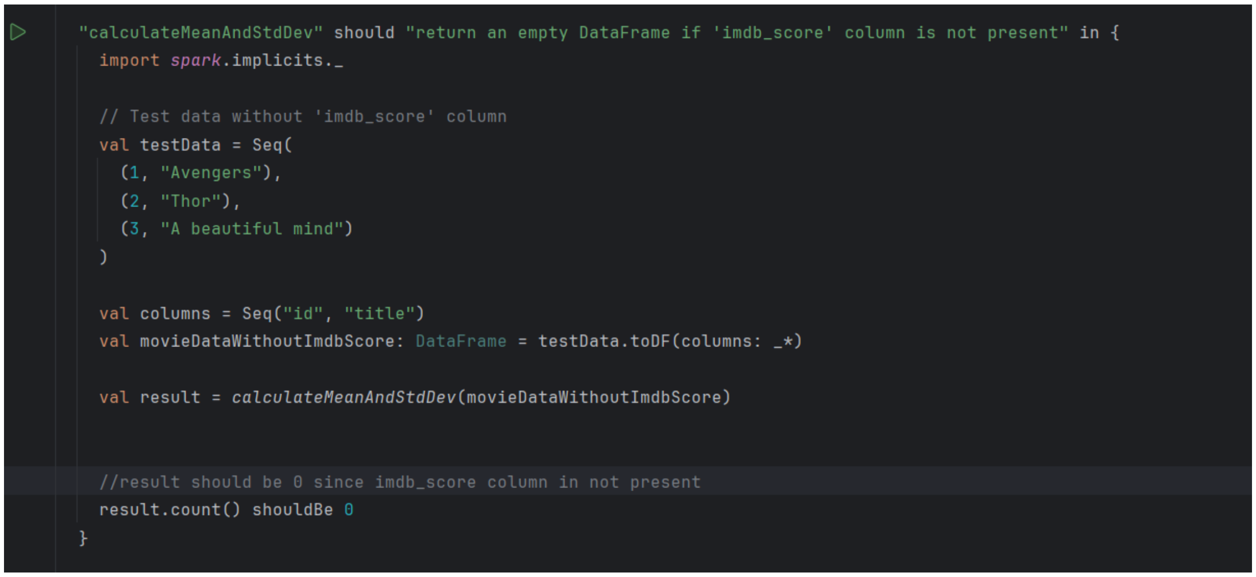
Output –



```
23/11/28 12:03:29 INFO TaskSchedulerImpl: Killing all running tasks in stage 3: Stage finished
23/11/28 12:03:29 INFO DAGScheduler: Job 2 finished: show at MovieRatingAnalyzer.scala:22, took 0.082598 s
23/11/28 12:03:29 INFO CodeGenerator: Code generated in 14.866796 ms
+-----------------+------------------+
|mean_rating      |std_dev_rating    |
+-----------------+------------------+
|6.453200745804848|0.9988071293753289|
+-----------------+------------------+
```

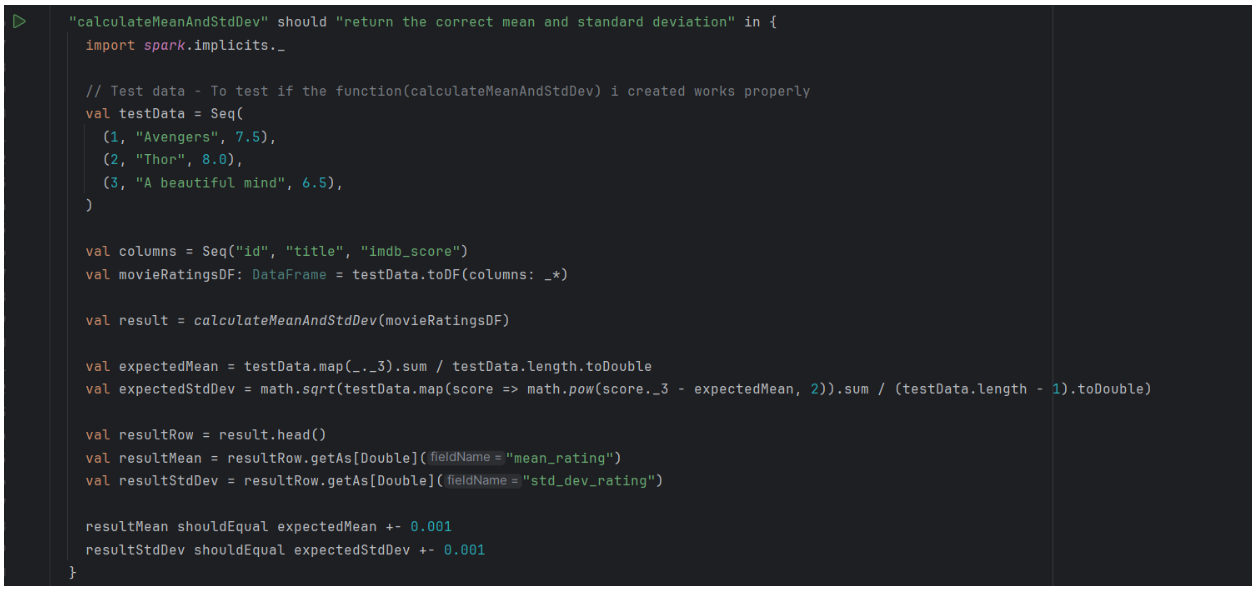
2) Test cases

Test case #1



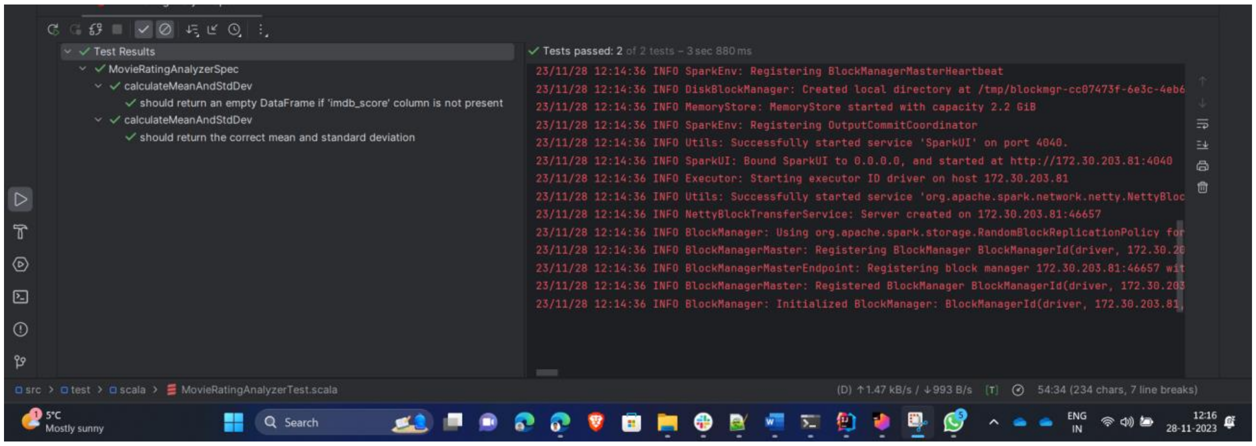
```scala
"calculateMeanAndStdDev" should "return an empty DataFrame if 'imdb_score' column is not present" in {
  import spark.implicits._

  // Test data without 'imdb_score' column
  val testData = Seq(
    (1, "Avengers"),
    (2, "Thor"),
    (3, "A beautiful mind")
  )

  val columns = Seq("id", "title")
  val movieDataWithoutImdbScore: DataFrame = testData.toDF(columns: _*)

  val result = calculateMeanAndStdDev(movieDataWithoutImdbScore)


  //result should be 0 since imdb_score column in not present
  result.count() shouldBe 0
}
```

Test case #2



```scala
"calculateMeanAndStdDev" should "return the correct mean and standard deviation" in {
  import spark.implicits._

  // Test data - To test if the function(calculateMeanAndStdDev) i created works properly
  val testData = Seq(
    (1, "Avengers", 7.5),
    (2, "Thor", 8.0),
    (3, "A beautiful mind", 6.5),
  )

  val columns = Seq("id", "title", "imdb_score")
  val movieRatingsDF: DataFrame = testData.toDF(columns: _*)

  val result = calculateMeanAndStdDev(movieRatingsDF)

  val expectedMean = testData.map(_._3).sum / testData.length.toDouble
  val expectedStdDev = math.sqrt(testData.map(score => math.pow(score._3 - expectedMean, 2)).sum / (testData.length - 1).toDouble)

  val resultRow = result.head()
  val resultMean = resultRow.getAs[Double]( fieldName = "mean_rating")
  val resultStdDev = resultRow.getAs[Double]( fieldName = "std_dev_rating")

  resultMean shouldEqual expectedMean +- 0.001
  resultStdDev shouldEqual expectedStdDev +- 0.001
}
```

Output



```
✓ Test Results
  ✓ MovieRatingAnalyzerSpec
    ✓ calculateMeanAndStdDev
      ✓ should return an empty DataFrame if 'imdb_score' column is not present
    ✓ calculateMeanAndStdDev
      ✓ should return the correct mean and standard deviation
```

```
✓ Tests passed: 2 of 2 tests – 3 sec 880 ms
23/11/28 12:14:36 INFO SparkEnv: Registering BlockManagerMasterHeartbeat
23/11/28 12:14:36 INFO DiskBlockManager: Created local directory at /tmp/blockmgr-cc07473f-6e3c-4eb6
23/11/28 12:14:36 INFO MemoryStore: MemoryStore started with capacity 2.2 GiB
23/11/28 12:14:36 INFO SparkEnv: Registering OutputCommitCoordinator
23/11/28 12:14:36 INFO Utils: Successfully started service 'SparkUI' on port 4040.
23/11/28 12:14:36 INFO SparkUI: Bound SparkUI to 0.0.0.0, and started at http://172.30.203.81:4040
23/11/28 12:14:36 INFO Executor: Starting executor ID driver on host 172.30.203.81
23/11/28 12:14:36 INFO Utils: Successfully started service 'org.apache.spark.network.netty.NettyBloc
23/11/28 12:14:36 INFO NettyBlockTransferService: Server created on 172.30.203.81:46657
23/11/28 12:14:36 INFO BlockManager: Using org.apache.spark.storage.RandomBlockReplicationPolicy for
23/11/28 12:14:36 INFO BlockManagerMaster: Registering BlockManager BlockManagerId(driver, 172.30.20
23/11/28 12:14:36 INFO BlockManagerMasterEndpoint: Registering block manager 172.30.203.81:46657 wit
23/11/28 12:14:36 INFO BlockManagerMaster: Registered BlockManager BlockManagerId(driver, 172.30.203
23/11/28 12:14:36 INFO BlockManager: Initialized BlockManager: BlockManagerId(driver, 172.30.203.81,
```

src > test > scala > MovieRatingAnalyzerTest.scala
(D) ↑1.47 kB/s / ↓993 B/s
54:34 (234 chars, 7 line breaks)
5°C Mostly sunny
Search
ENG IN
12:16 28-11-2023