NAME: MARIA GLORIA OBONO ONDO

NUID: 002667315

Assignment 7: Analizing Movie Rating

I got the ratings_small.csv file from: https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset?select=ratings.csv

CODE: MovieratingDataset.scala

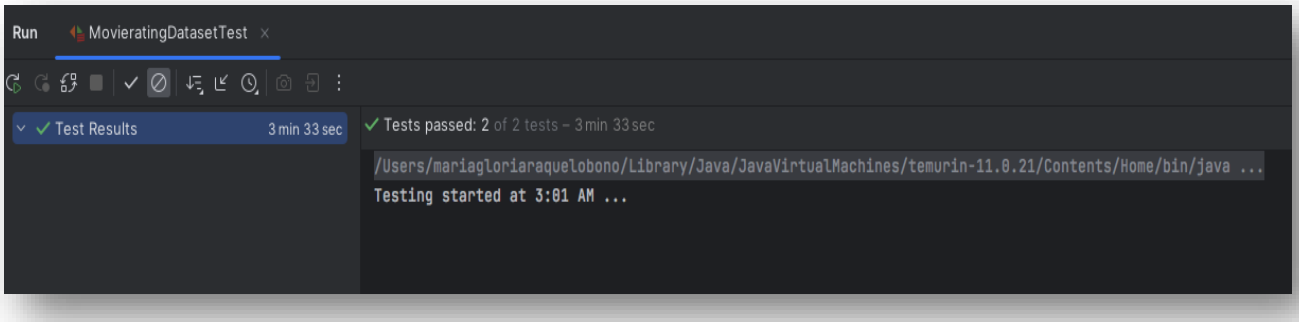
```
import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.functions.{mean, stddev}

object MovieratingDataset {
  def main(args: Array[String]): Unit = {
    // Create a SparkSession
    val spark = SparkSession.builder()
      .appName("MovieratingDataset")
      .master("local[*]") // Change this to your cluster setup
      .getOrCreate()

    // Read the CSV file into a DataFrame
    val filePath = "/Users/mariagloriaraquelobono/Fall2023/Movie-
Analyzer/src/main/resources/ratings_small.csv"
    val movieData = spark.read.option("header", "true").csv(filePath)

    // Calculate mean rating and standard deviation
    val meanRating = BigDecimal(movieData.select(mean("rating")).first().getDouble(0)).setScale(3,
BigDecimal.RoundingMode.HALF_UP).toDouble
    val stdDevRating = BigDecimal(movieData.select(stddev("rating")).first().getDouble(0)).setScale(3,
BigDecimal.RoundingMode.HALF_UP).toDouble

    println(s"Mean Rating: $meanRating")
    println(s"Standard Deviation of Rating: $stdDevRating")

    // Stop the SparkSession
    spark.stop()
  }

}
```

```
Mean Rating: 3.544
Standard Deviation of Rating: 1.058
```

CODE: MovieratingDatasetTest.scala

```
import org.apache.spark.sql.{SparkSession, DataFrame}
import org.apache.spark.sql.functions.{mean, stddev}
import org.scalatest.funsuite.AnyFunSuite

class MovieratingDatasetTest extends AnyFunSuite {

  def readRatingsCSV(spark: SparkSession, filePath: String): DataFrame = {
    // Read ratings.csv
    spark.read.option("header", "true").csv(filePath)
  }


  test("Test movie ratings analysis with merged data") {
    val spark = SparkSession.builder()
      .appName("MovieratingDatasetTest")
      .master("local[*]")
      .getOrCreate()

    import spark.implicits._

    // Replace these paths with your actual file paths

    val ratingFilePath = "/Users/mariagloriaraquelobono/Fall2023/Movie-
Analyzer/src/main/resources/ratings_small.csv"

    val ratingsData = readRatingsCSV(spark, ratingFilePath)

    val calculatedStats = ratingsData.agg(mean("rating").as("MeanRating"),
stddev("rating").as("StdDevRating")).head()
    val meanRating = BigDecimal(calculatedStats.getAs[Double]("MeanRating")).setScale(3,
BigDecimal.RoundingMode.HALF_UP).toDouble
    val stdDevRating = BigDecimal(calculatedStats.getAs[Double]("StdDevRating")).setScale(3,
BigDecimal.RoundingMode.HALF_UP).toDouble
```

```
    // Define expected values based on your test data
    val meanRatingExpected = 3.544
    val stdDevRatingExpected = 1.058

    assert(meanRating === meanRatingExpected)
    assert(stdDevRating === stdDevRatingExpected)

    spark.stop()
  }

  test("Test handling of empty ratings dataset") {
    val spark = SparkSession.builder()
      .appName("MovieratingDatasetTest")
      .master("local[*]")
      .getOrCreate()

    import spark.implicits._

    // Create an empty DataFrame to simulate an empty ratings dataset
    val emptyTestData = Seq.empty[(String, Double)].toDF("userId", "rating")

    val calculatedStats = emptyTestData.agg(mean("rating").as("MeanRating"),
stddev("rating").as("StdDevRating")).head()
    val meanRating = calculatedStats.getAs[Double]("MeanRating")
    val stdDevRating = calculatedStats.getAs[Double]("StdDevRating")

    // Define expected values for an empty dataset
    val meanRatingExpected = 0.0 // Expected mean rating for an empty dataset
    val stdDevRatingExpected = 0.0 // Expected standard deviation for an empty dataset

    assert(meanRating === meanRatingExpected)
    assert(stdDevRating === stdDevRatingExpected)

    spark.stop()
  }

}
```


Run
MovieratingDatasetTest
Test Results
3 min 33 sec
Tests passed: 2 of 2 tests – 3 min 33 sec
/Users/mariagloriaraquelobono/Library/Java/JavaVirtualMachines/temurin-11.0.21/Contents/Home/bin/java ...
Testing started at 3:01 AM ...