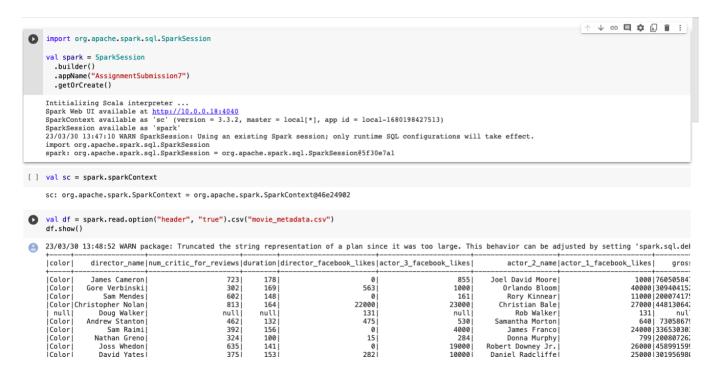
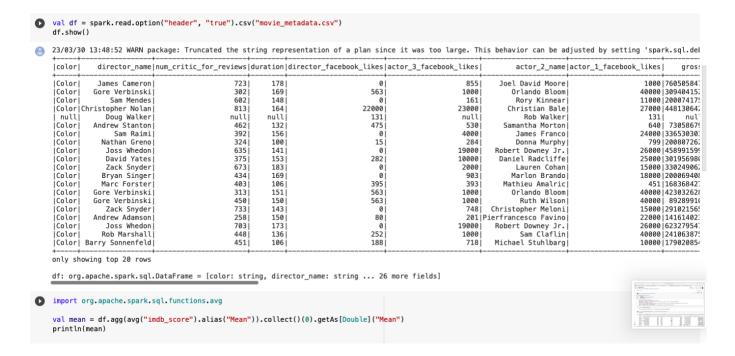
Komal Suryan: NUID 002747707

Big Data System Engineering with Scala Spring 2023 Assignment No. 7 (Spark)



You are required to analyze a movie rating dataset. The data is stored in a CSV file (either use the one in the repository or download the latest from Kaggle). You need to read this file into spark and calculate the mean rating and standard deviation for all movies. There is no test case provided for you, so you need to write your own test cases to ensure that at least your program works well.





Unit Tests:

Test case 1: When the column name is valid and contains numerical values, the method should return the correct average of the column.

Test case 2: When the column name is valid but contains non-numerical values, the method should return null.

Test case 3: When the column name is valid and contains numerical values, the method should return the correct standard deviation of the column.

• 6.453200745804848
import org.apache.spark.sql.functions.avg
mean: Double = 6.4532007458084848

[] import org.apache.spark.sql.functions.stddev

val stdDev = df.agg(stddev("imdb_score").alias("Standard Deviation")).collect()(0).getAs[Double]("Standard Deviation")

println(stdDev)

0.9088071293753289
import org.apache.spark.sql.functions.stddev
stdDev: bouble = 0.9988071293753289

import org.apache.spark.sql.functions.{avg, stddev}

// Test case 1: When the column name is valid and contains numerical values, the val df1 = Seq([1, 5.5), (2, 4.5), (3, 6.5).toDF("id", "value")
assert(df1.agg(avg("value")).head().getDouble(0) == 5.5)

// Test case 2: When the column name is valid but contains non-numerical values, val df2 = Seq([1, "foo"), (2, "bar"), (3, "baz")).toDF("id", "value")
assert(df2.agg(avg("value")).head().isNullAt(0))

// Test case 3: When the column name is valid and contains numerical values, the val df3 = Seq([1, 5.5), (2, 4.5), (3, 6.5)).toDF("id", "value")
assert(df3.agg(stddev("value")).head().getDouble(0) == 1.0)

// Test case 3: When the column name is valid and contains numerical values, the val df3 = Seq([1, 5.5), (2, 4.5), (3, 6.5)).toDF("id", "value")
assert(df3.agg(stddev("value")).head().getDouble(0) == 1.0)

// Test case 3: When the column name is valid and contains numerical values, the valid of assert(df3.agg(stddev("value")).head().getDouble(0) == 1.0)

// Test case 3: When the column name is valid and contains numerical values, the valid of assert(df3.agg(stddev("value")).head().getDouble(0) == 1.0)

// Test case 3: When the column name is valid and contains numerical values, the valid of assert(df3.agg(stddev("value")).head().getDouble(0) == 1.0)

// Test case 3: When the column name is valid and contains numerical values, the valid of assert(df3.agg(stddev("value")).head().getDouble(0) == 1.0)

// Test case 3: When the column name is valid and contains numerical values, the valid of assert(df3.agg(stddev("value")).head().getDouble(0) == 1.0)

// Test case 3: When the column n

import org.apache.spark.sql.functions.{avg, stddev}
df1: org.apache.spark.sql.DataFrame = [id: int, value: double]
df2: org.apache.spark.sql.DataFrame = [id: int, value: string]
df3: org.apache.spark.sql.DataFrame = [id: int, value: double]