Ebenezer Ajay Williams – SEC01 (NUID 002166250)

Big Data System Engineering with Scala
Spring 2022
Assignment No. #6 Analyzing Movie Rating

Task

- **Created a repo for Assignment 6**
- **Used Spark Sql to read data from csv.**
- **Calculated the mean and standard deviation of the imdb_score column**
- **Wrote unit tests to test the application.**

Source of Dataset: Class Repository

github commit : https://github.com/ebiskhan123/Scala-Spark-assignment

Solution/ Unit test (screenshot)

MovieAnalysis class

```scala
object MovieAnalysis extends App {

  val spark: SparkSession = SparkSession
    .builder()
    .appName( name = "MovieAnalysis")
    .master( master = "local[*]")
    .getOrCreate()

  spark.sparkContext.setLogLevel("WARN")
  val path = "/Users/ebby/Documents/Grad/Scala/src/main/resources/movie_metadata.csv";
  val df = spark.read.option("header", "true").csv(path)
  df.show( numRows = 3)
  ProcessData.findMean(df, col = "imdb_score")show()
  ProcessData.findStDev(df, col = "imdb_score")show()

}

object ProcessData {
  def findMean( df:DataFrame,  col: String):DataFrame = {
    df.select(avg(col))
  }

  def findStDev( df:DataFrame,  col: String):DataFrame = {
    df.select(stddev_pop(col))
  }
}
```

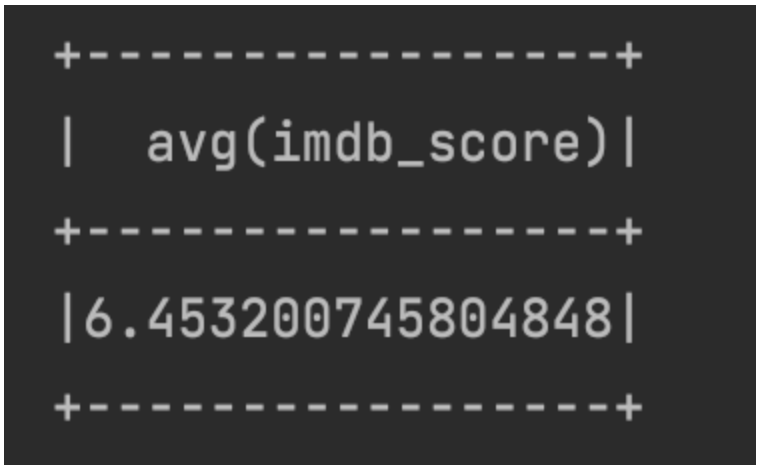
MovieAnalysisTest

```scala
class MovieAnalysisTest extends AnyFlatSpec with Matchers {

  behavior of "test of 20 rows"
  it should "test movie_metadata.csv" in {
    val spark: SparkSession = SparkSession
      .builder()
      .appName( name = "MovieAnalysis")
      .master( master = "local[*]")
      .getOrCreate()

    spark.sparkContext.setLogLevel("WARN")
    val path = "/Users/ebby/Documents/Grad/Scala/src/main/resources/movie_metadata.csv"
    val df = spark.read.option("header", "true").csv(path).limit(20)
    ProcessData.findMean(df, col = "imdb_score").first().getDouble(0) shouldBe(7.095000000000001)
    ProcessData.findStDev(df, col = "imdb_score").first().getDouble(0) shouldBe(0.6192535829528967)
  }


  behavior of "test the mean and std dev for the entire dataset"
  it should "test movie_metadata.csv" in {
    val spark: SparkSession = SparkSession
      .builder()
      .appName( name = "MovieAnalysis")
      .master( master = "local[*]")
      .getOrCreate()

    spark.sparkContext.setLogLevel("WARN")
    val path = "/Users/ebby/Documents/Grad/Scala/src/main/resources/movie_metadata.csv"
    val df = spark.read.option("header", "true").csv(path)
    ProcessData.findMean(df, col = "imdb_score").first().getDouble(0) shouldBe(6.453200745804848)
    ProcessData.findStDev(df, col = "imdb_score").first().getDouble(0) shouldBe(0.9984966998015917)
  }
```

Top 3 rows of the dataset

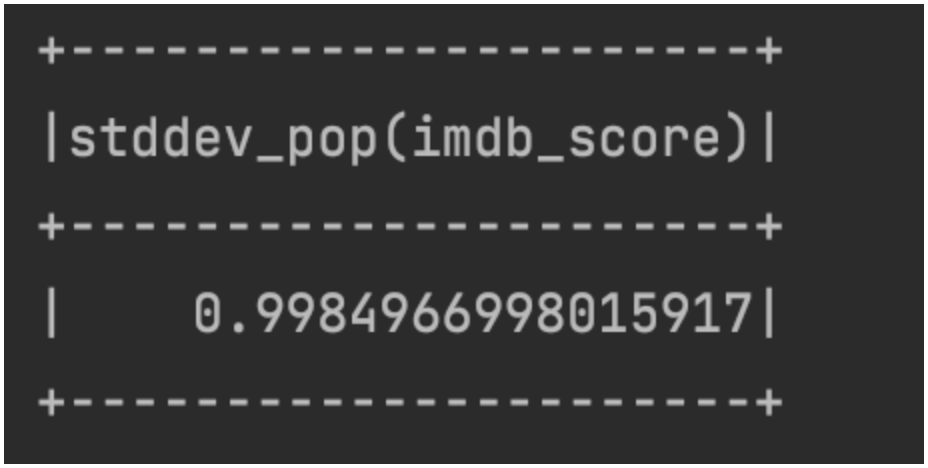
color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_name	actor_1_facebook_likes	gross	genre
Color	James Cameron	723	178	0	855	Joel David Moore	1000	760505847	Action|Adventure|..
Color	Gore Verbinski	302	169	563	1000	Orlando Bloom	40000	309404152	Action|Adventure|..
Color	Sam Mendes	602	148	0	161	Rory Kinnear	11000	200074175	Action|Adventure|..

Mean IMDB score



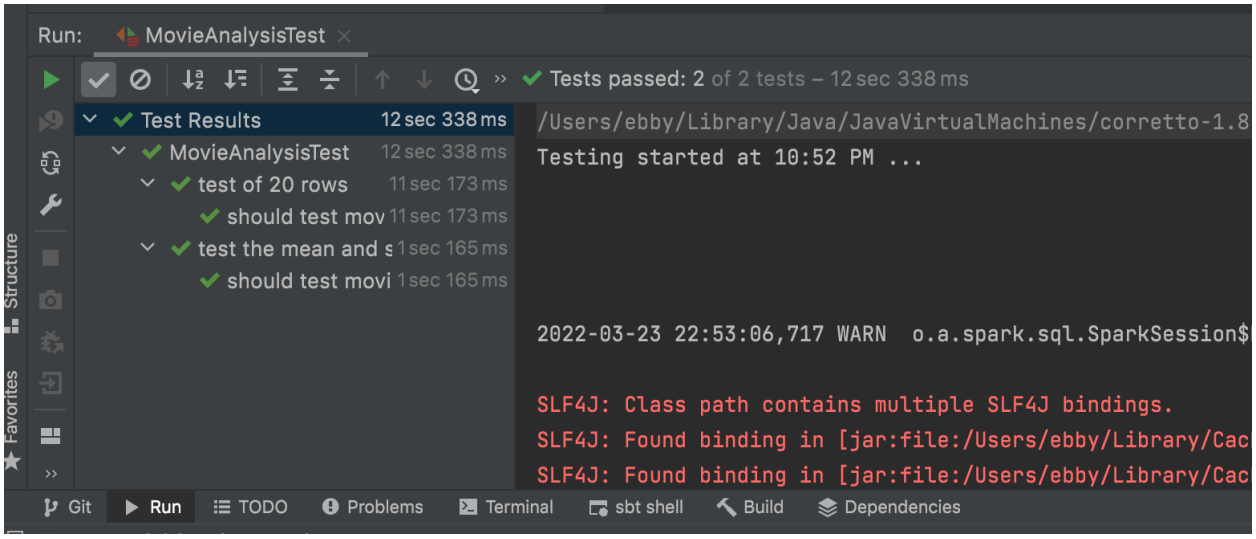
```
+------------------+
|  avg(imdb_score)|
+------------------+
|6.453200745804848|
+------------------+
```

Standard Deviation



```
+----------------------+
|stddev_pop(imdb_score)|
+----------------------+
|    0.9984966998015917|
+----------------------+
```

Unit test results:



Result:

All unit tests have run successfully

Mean imdb score : `6.453200745804848`

Standard Deviation of imdb score : `0.9984966998015917`