Anirudha Joshi – SEC01 (NUID 002991365)

Big Data System Engineering with Scala
Spring 2023
Assignment No. 1 (Spark)

GitHub - https://github.com/anirudhajoshi2808/Anirudha_Joshi_CSYE7200

- Questions:

You are to load the dataset using Spark and perform the operations to answer the following questions.

1) What is the average ticket fare for each Ticket class?

(The ticket class is the pclass column. 1st = Upper; 2nd = Middle ; 3rd = Lower. Find the average cost of the ticket fare for the 1st class, the 2ns class and the 3rd class)

2) What is the survival percentage for each Ticket class? Which class has the highest survival rate?

(survival rate = number of survived per class/ total number of passengers . In the survival column, 0 = dead and 1 = survived)

3) Rose DeWitt Bukater was 17 years old when she boarded the titanic. She is traveling with her mother and fiance(they are not married yet, so they are not related). She is traveling first class. With the information of her age, gender, class she is traveling in, and the fact that she is traveling with one parent, find the number of passengers who could possibly be Rose. (PS: if you watched the movie you will know if she survived or died)

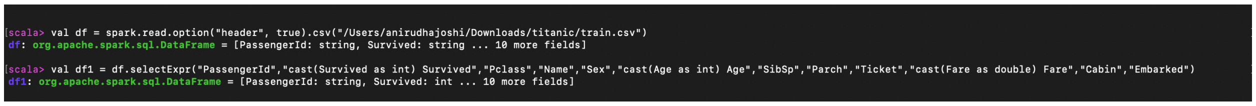
4) Jack Dawson born in 1892 died on April 15, 1912. He is either 20 or 19 years old. He travels 3rd class and has no relatives onboard. Find the number of passengers who could possibly be Jack? (PS: Yeah he's the guy who gets Rose)

5) Split the age for every 10 years. 1-10 as one age group, 11- 20 as another etc.

What is the relation between the ages and the ticket fare? Which age group most likely survived ?

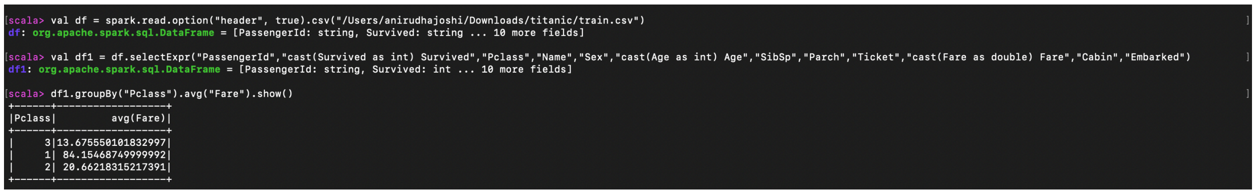
Code

1. Setup:



```
[scala> val df = spark.read.option("header", true).csv("/Users/anirudhajoshi/Downloads/titanic/train.csv")
 df: org.apache.spark.sql.DataFrame = [PassengerId: string, Survived: string ... 10 more fields]

[scala> val df1 = df.selectExpr("PassengerId","cast(Survived as int) Survived","Pclass","Name","Sex","cast(Age as int) Age","SibSp","Parch","Ticket","cast(Fare as double) Fare","Cabin","Embarked")
 df1: org.apache.spark.sql.DataFrame = [PassengerId: string, Survived: int ... 10 more fields]
```

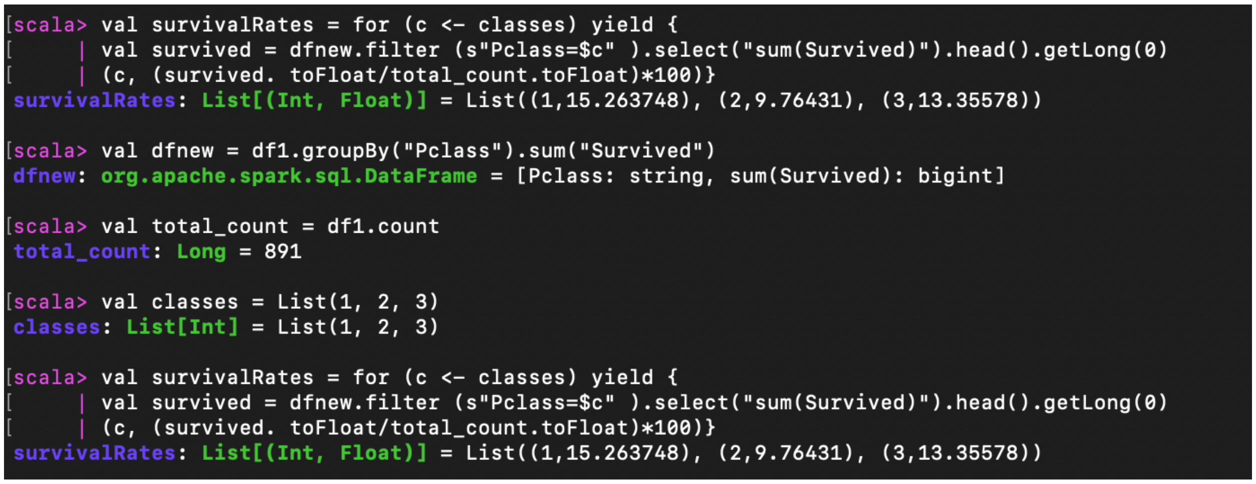
- **Answer 1**



```
[scala> val df = spark.read.option("header", true).csv("/Users/anirudhajoshi/Downloads/titanic/train.csv")
 df: org.apache.spark.sql.DataFrame = [PassengerId: string, Survived: string ... 10 more fields]

[scala> val df1 = df.selectExpr("PassengerId","cast(Survived as int) Survived","Pclass","Name","Sex","cast(Age as int) Age","SibSp","Parch","Ticket","cast(Fare as double) Fare","Cabin","Embarked")
 df1: org.apache.spark.sql.DataFrame = [PassengerId: string, Survived: int ... 10 more fields]

[scala> df1.groupBy("Pclass").avg("Fare").show()
 +------+------------------+
 |Pclass|         avg(Fare)|
 +------+------------------+
 |     3|13.675550101832997|
 |     1| 84.15468749999992|
 |     2| 20.66218315217391|
 +------+------------------+
```

- **Answer 2**

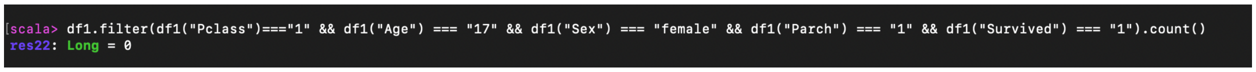
Class 1 has the highest survival rate of 15.26%



```
[scala> val survivalRates = for (c <- classes) yield {
[     | val survived = dfnew.filter (s"Pclass=$c" ).select("sum(Survived)").head().getLong(0)
[     | (c, (survived. toFloat/total_count.toFloat)*100)}
 survivalRates: List[(Int, Float)] = List((1,15.263748), (2,9.76431), (3,13.35578))

[scala> val dfnew = df1.groupBy("Pclass").sum("Survived")
 dfnew: org.apache.spark.sql.DataFrame = [Pclass: string, sum(Survived): bigint]

[scala> val total_count = df1.count
 total_count: Long = 891

[scala> val classes = List(1, 2, 3)
 classes: List[Int] = List(1, 2, 3)

[scala> val survivalRates = for (c <- classes) yield {
[     | val survived = dfnew.filter (s"Pclass=$c" ).select("sum(Survived)").head().getLong(0)
[     | (c, (survived. toFloat/total_count.toFloat)*100)}
 survivalRates: List[(Int, Float)] = List((1,15.263748), (2,9.76431), (3,13.35578))
```

- **Answer 3**

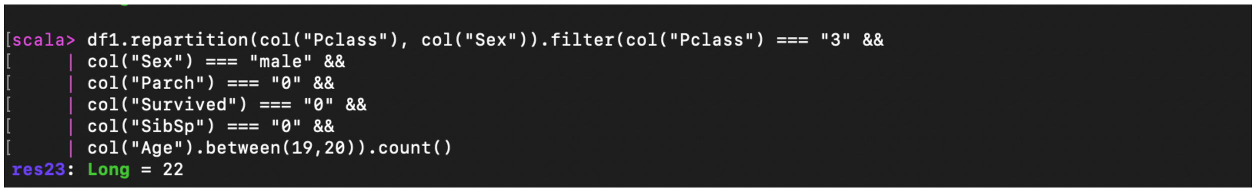
There are 0 passengers who could possibly be Rose



```
[scala> df1.filter(df1("Pclass")==="1" && df1("Age") === "17" && df1("Sex") === "female" && df1("Parch") === "1" && df1("Survived") === "1").count()
res22: Long = 0
```

- **Answer 4**

  There are 22 passengers who could be Jack



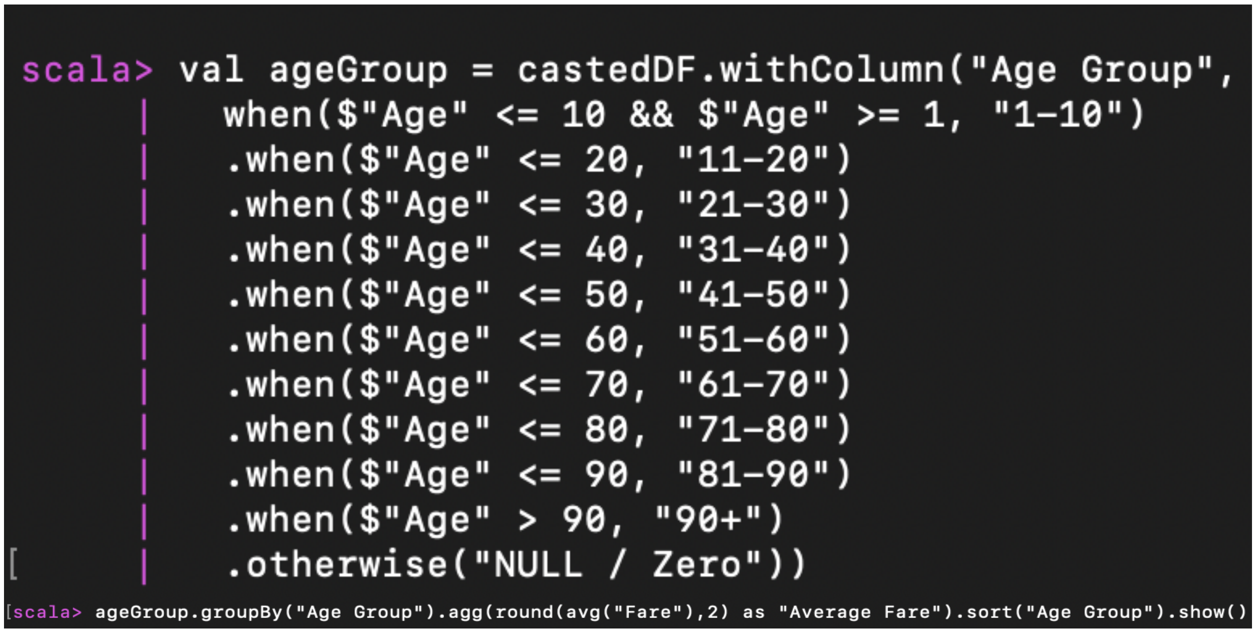
```
[scala> df1.repartition(col("Pclass"), col("Sex")).filter(col("Pclass") === "3" &&
[     | col("Sex") === "male" &&
[     | col("Parch") === "0" &&
[     | col("Survived") === "0" &&
[     | col("SibSp") === "0" &&
[     | col("Age").between(19,20)).count()
 res23: Long = 22
```

- **Answer 5:**

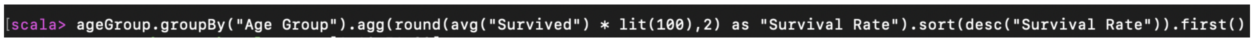
  The age group 31 to 70 spent more on ticket fares compared to

  other age groups.

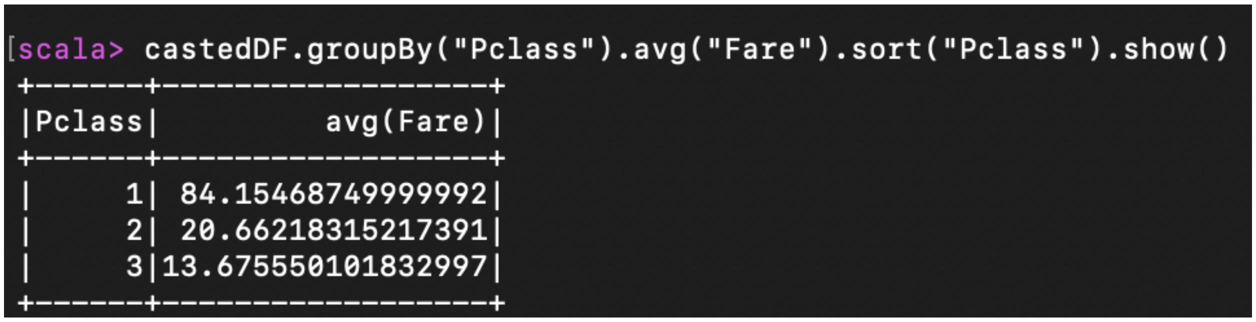
  Age group 1-10 had the highest survival rate of 54.39%



```
scala> val ageGroup = castedDF.withColumn("Age Group",
     |   when($"Age" <= 10 && $"Age" >= 1, "1-10")
     |   .when($"Age" <= 20, "11-20")
     |   .when($"Age" <= 30, "21-30")
     |   .when($"Age" <= 40, "31-40")
     |   .when($"Age" <= 50, "41-50")
     |   .when($"Age" <= 60, "51-60")
     |   .when($"Age" <= 70, "61-70")
     |   .when($"Age" <= 80, "71-80")
     |   .when($"Age" <= 90, "81-90")
     |   .when($"Age" > 90, "90+")
[    |   .otherwise("NULL / Zero"))
[scala> ageGroup.groupBy("Age Group").agg(round(avg("Fare"),2) as "Average Fare").sort("Age Group").show()
```



```
[scala> ageGroup.groupBy("Age Group").agg(round(avg("Survived") * lit(100),2) as "Survival Rate").sort(desc("Survival Rate")).first()
```



```
[scala> castedDF.groupBy("Pclass").avg("Fare").sort("Pclass").show()
+------+------------------+
|Pclass|         avg(Fare)|
+------+------------------+
|     1| 84.15468749999992|
|     2| 20.66218315217391|
|     3|13.675550101832997|
+------+------------------+
```