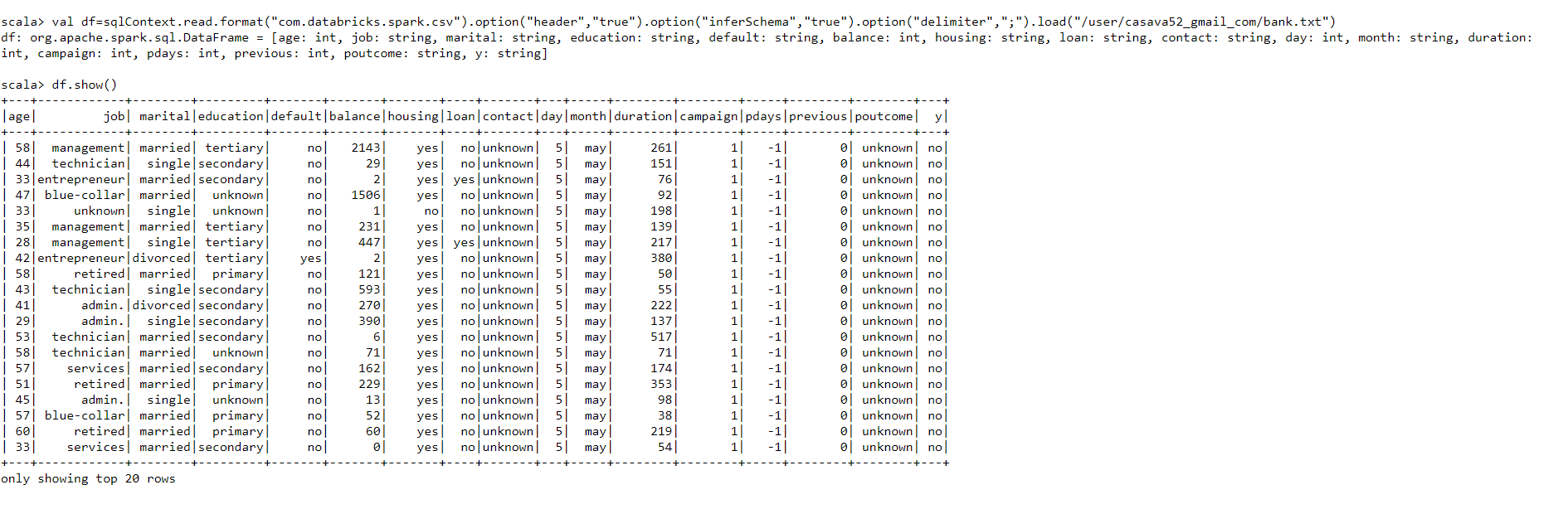
Nirmal Patel Portuguese Bank Subscription Data Big Data Scala

1.Load data and create Spark data frame

val df=sqlContext.read.format("com.databricks.spark.csv").option("header","true").option("inferSchema","true").option("delimiter",";").load("/user/casava52\_gmail\_com/[bank.txt](http://sl.cloudloka.com:8888/hue/filebrowser/view=/user/casava52_gmail_com/bank.txt)")

df.show()

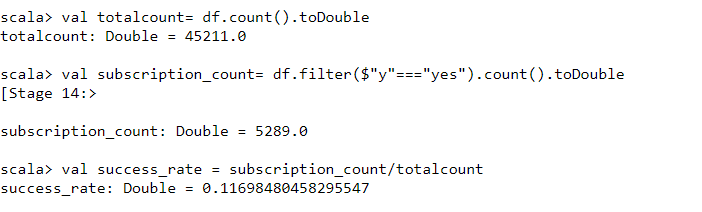


2. Give marketing success rate. (No. of people subscribed / total no. of entries)

val totalcount= df.count().toDouble =45211

val subscription\_count=df.filter($”y”===”yes”).count().toDouble = 5289

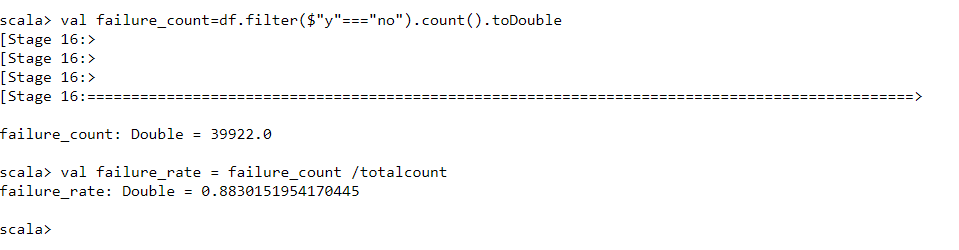
val success\_rate = subscription\_count/totalcount =0.11



2a Give marketing failure rate

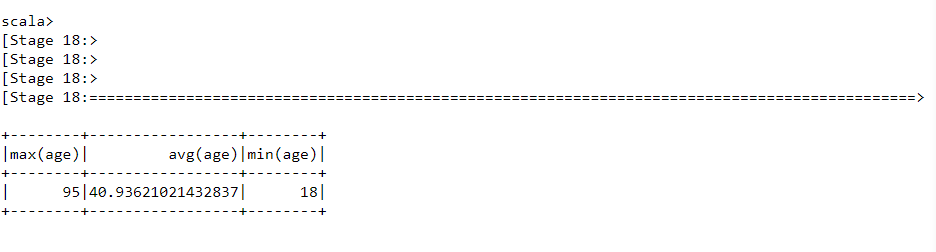
val failure\_count=df.filter($”y”===”no”).count().toDouble =39922

val failure\_rate = failure\_count /totalcount =0.88



3. Maximum, Mean, and Minimum age of average targeted customer

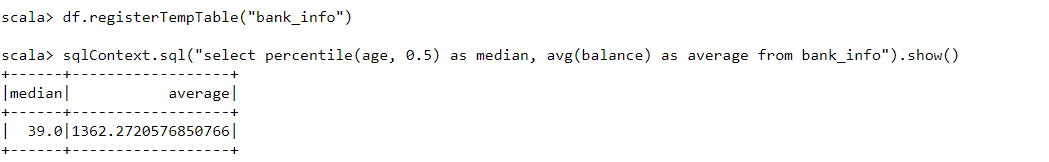
df.select(max($”age”),avg($”age”),min($”age”)).show()



4. Check quality of customers by checking average balance, median balance of customers

df.registerTempTable("bank\_info")

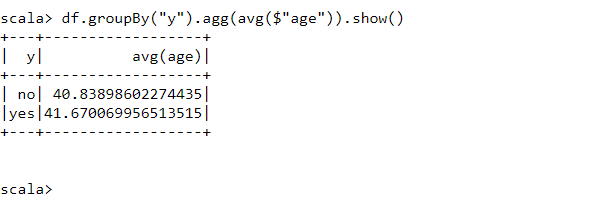
sqlContext.sql("select percentile(age, 0.5) as median, avg(balance) as average from bank\_info").show()



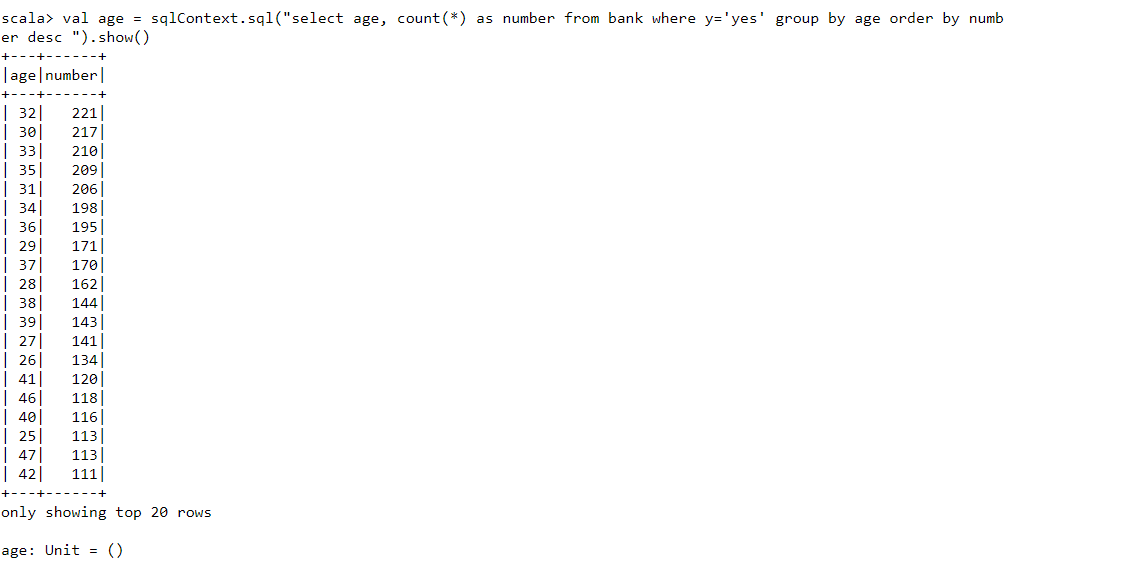
5. Check if age matters in marketing subscription for deposit

Group by y average of age said y and said no aggregate average on g

Df.groupBy(“y”).agg(avg($”age”)).show()



val age = sqlContext.sql("select age, count(\*) as number from bank where y='yes' group by age order by number desc ").show()



scala> val age = sqlContext.sql("select age, count(\*) as number from bank where y='yes' group by age order by numb

er desc ").show()

+---+------+

|age|number|

+---+------+

| 32| 221|

| 30| 217|

| 33| 210|

| 35| 209|

| 31| 206|

| 34| 198|

| 36| 195|

| 29| 171|

| 37| 170|

| 28| 162|

| 38| 144|

| 39| 143|

| 27| 141|

| 26| 134|

| 41| 120|

| 46| 118|

| 40| 116|

| 25| 113|

| 47| 113|

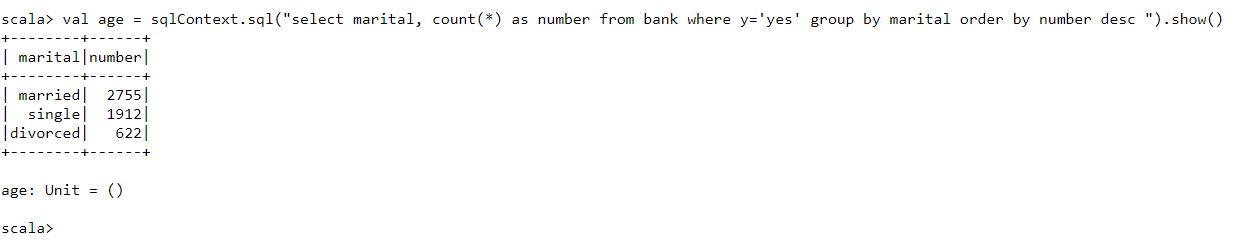
| 42| 111|

+---+------+

only showing top 20 rows

6. Check if marital status mattered for subscription to deposit.

val age = sqlContext.sql("select marital, count(\*) as number from bank where y='yes' group by marital order by number desc ").show()



+--------+------+

| marital|number|

+--------+------+

| married| 2755|

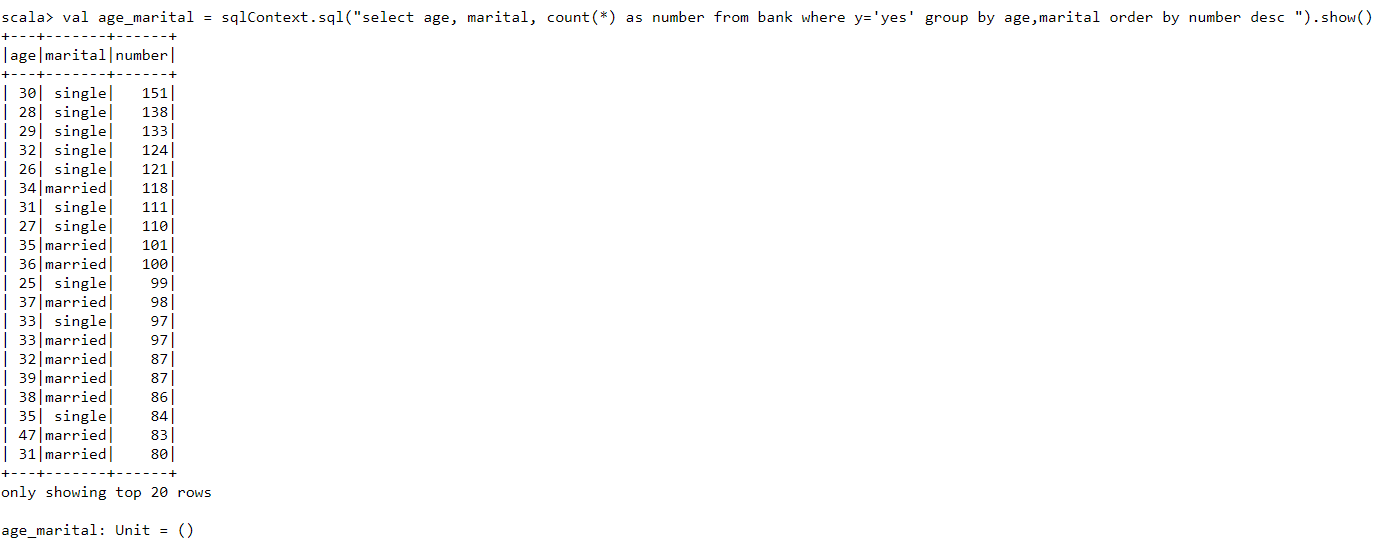
| single| 1912|

|divorced| 622|

+--------+------+

7. Check if age and marital status together mattered for subscription to deposit scheme

val age\_marital = sqlContext.sql("select age, marital, count(\*) as number from bank where y='yes' group by age,marital order by number desc ").show()



+---+-------+------+

|age|marital|number|

+---+-------+------+

| 30| single| 151|

| 28| single| 138|

| 29| single| 133|

| 32| single| 124|

| 26| single| 121|

| 34|married| 118|

| 31| single| 111|

| 27| single| 110|

| 35|married| 101|

| 36|married| 100|

| 25| single| 99|

| 37|married| 98|

| 33| single| 97|

| 33|married| 97|

| 32|married| 87|

| 39|married| 87|

| 38|married| 86|

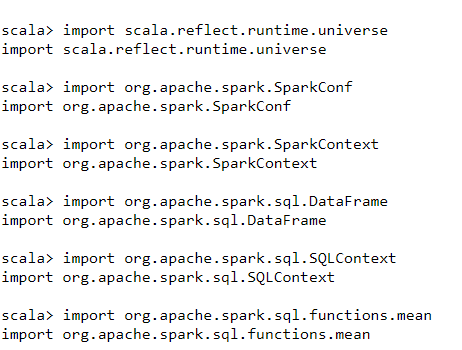
| 35| single| 84|

| 47|married| 83|

| 31|married| 80|

+---+-------+------+

8. Do feature engineering for column—age and find right age effect on campaign



scala> import scala.reflect.runtime.universe

import scala.reflect.runtime.universe

scala> import org.apache.spark.SparkConf

import org.apache.spark.SparkConf

scala> import org.apache.spark.SparkContext

import org.apache.spark.SparkContext

scala> import org.apache.spark.sql.DataFrame

import org.apache.spark.sql.DataFrame

scala> import org.apache.spark.sql.SQLContext

import org.apache.spark.sql.SQLContext

scala> import org.apache.spark.sql.functions.mean

import org.apache.spark.sql.functions.mean

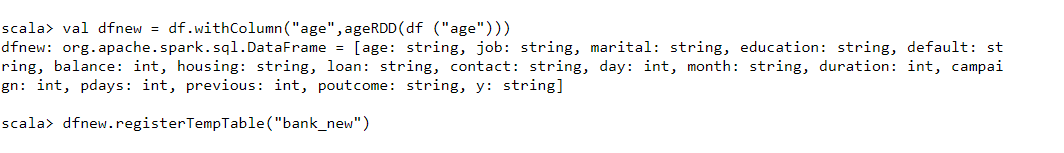
val ageRDD = sqlContext.udf.register("ageRDD",(age:Int) => { if (age > 17 && age <= 43) "Youngest 1/3 sample (18-43 yrs)" else if (age > 44 && age <= 69) "Middle 1/3 sample (44-69 yrs)" else "Oldest 1/3 of sample (70-95yrs)" })

18-43 Youngest 1/3 sample (18-43 yrs)

44-69 Middle 1/3 sample (44-69 yrs)

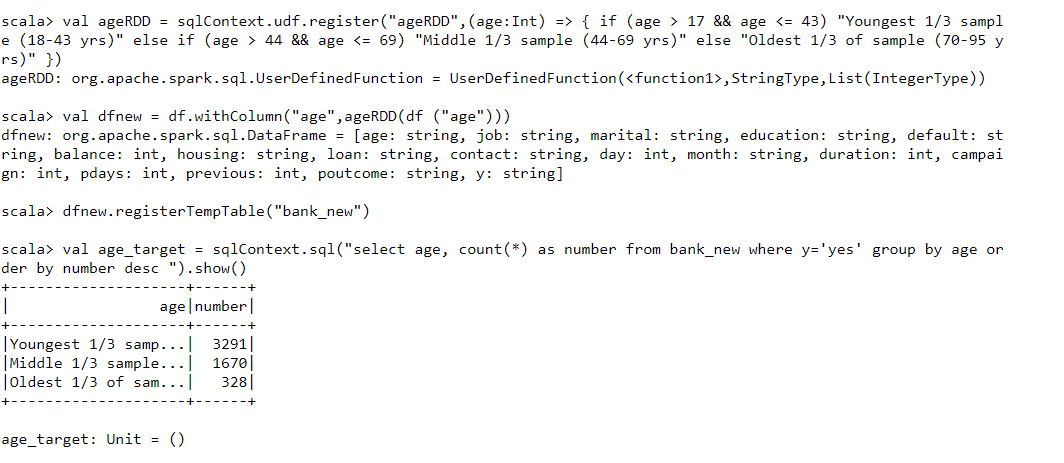
70-95 Oldest 1/3 of sample (70-95yrs)

val dfnew = df.withColumn("age",ageRDD(df ("age")))



dfnew.registerTempTable("bank\_new")

val age\_target = sqlContext.sql("select age, count(\*) as number from bank\_new where y='yes' group by age order by number desc ").show()



scala> val age\_target = sqlContext.sql("select age, count(\*) as number from bank\_new where y='yes' group by age or

der by number desc ").show()

+--------------------+------+

| age|number|

+--------------------+------+

|Youngest 1/3 samp...| 3291|

|Middle 1/3 sample...| 1670|

|Oldest 1/3 of sam...| 328|

+--------------------+------+

age\_target: Unit = ()