1. Begin by dropping all the double from csv file.

Load data and create Spark data frame

val sqlContext = new org.apache.spark.sql.SQLContext(sc)

val bankDf = spark.read.option("header","true").option("delimiter",";").csv("gs://hdpuserbucket/bankdata.csv")

bankDf.registerTempTable("bankdata")

1. Marketing success rate. (No. of people subscribed / total no. of entries)

val totalentries = sqlContext.sql("select count(\*) from bankdata")

val noofsubscribed = sqlContext.sql("select count(\*) from bankdata where y='yes'")

val Marketing\_success\_rate = noofsubscribed.head().getLong(0).toDouble / totalentries.head().getLong(0).toDouble

2a. val Marketing\_failure\_rate = 100 - Marketing\_success\_rate

1. Maximun and Minimum and Mean

val min\_max\_avg\_age = bankDf.agg(min(“age”), max(“age”),avg(“age”)).show

1. Average Balance and Median Balance

val avg\_balance = bankDf.agg(avg("balance")).show

val median\_balance = sqlContext.sql("select percentile(balance,0.5) from bankdata")

1. bankDf.groupBy("y").agg(avg("age")).show

Yes, Age matters in marketing subscription for deposit.

1. bankDf.filter($"y"==="yes").groupBy("marital").count.show

Marketing subscription for deposit is high for people whose marital status

is married.

1. sqlContext.sql("select age,marital,count(\*) from bankdata where y='yes' group by marital,age").show
2. def ageToCategory = udf((age:Int) => {

age match {

case t if t < 30 => "Young"

case t if t > 65 => "Old"

case \_ => "Mid Age"

}})

val newbankDf = bankDf.withColumn("agecategory",agetocategory(bankDf("age")))

newbankDf.filter($"y"==="yes").groupBy("agecategory").count.show

Mid-Age people subscribe the most.