**Big Data Hadoop and Spark Developer**

**Project 4**

**Market Analysis in Banking Domain**

**Made By:**

**Siddhartha Patra**

**Background and Objective:**

Your client, a Portuguese banking institution, ran a marketing campaign to convince potential customers to invest in a bank term deposit scheme.

The marketing campaigns were based on phone calls. Often, the same customer was contacted more than once through phone, in order to assess if they would want to subscribe to the bank term deposit or not. You have to perform the marketing analysis of the data generated by this campaign.

**Analysis tasks to be done:**

The data size is huge and the marketing team has asked you to perform the below analysis-

1. Load data and create a Spark data frame
2. Give marketing success rate (No. of people subscribed / total no. of entries)
3. Give marketing failure rate
4. Give the maximum, mean, and minimum age of the average targeted customer
5. Check the quality of customers by checking average balance, median balance of customers
6. Check if age matters in marketing subscription for deposit
7. Check if marital status mattered for a subscription to deposit
8. Check if age and marital status together mattered for a subscription to deposit scheme
9. Do feature engineering for the bank and find the right age effect on the campaign.

**Source Code (Scala)**

import org.apache.spark.sql.\_

import org.apache.spark.sql.types.\_

import sqlContext.implicits.\_

val data = sc.textFile("bank-full.csv").map(x => x.split(";(?=([^\"]\*\"[^\"]\*\")\*[^\"]\*$)",-1))

val header = data.first()

val filtered = data.filter(x => x(0)!= header(0))

val rdds = filtered.map(x => Row(x(0).toInt, x(1),x(2),x(3),x(4), x(5).toInt,x(6),x(7),x(8), x(9).toInt,x(10),x(11).toInt,x(12).toInt, x(13).toInt,x(14).toInt,x(15),x(16) ))

val schema = StructType( List(StructField("age", IntegerType, true),StructField("job", StringType, true) ,StructField("marital", StringType, true),StructField("education", StringType, true) ,StructField("default", StringType, true),StructField("balance", IntegerType, true) ,StructField("housing", StringType, true) ,StructField("loan", StringType, true) ,StructField("contact", StringType, true) ,StructField("day", IntegerType, true) ,StructField("month", StringType, true) ,StructField("duration", IntegerType, true) ,StructField("campaign", IntegerType, true) ,StructField("pdays", IntegerType, true) ,StructField("previous", IntegerType, true) ,StructField("poutcome", StringType, true) ,StructField("y", StringType, true)) )

val df = sqlContext.createDataFrame(rdds, schema)

val success\_rate = (df.filter($"y" === "\"yes\"").count).toDouble / (df.count).toDouble

df.select(max($"age"), min($"age") , mean($"age")).show

sqlContext.sql("select percentile(age, 0.50) from df").show

sqlContext.sql("select max(age), min(age), avg(age) , percentile(age, 0.50) from df").show

sqlContext.sql("select avg(balance), percentile(balance, 0.50) from df").show

df.groupBy("y").agg(avg($"age")).show

df.groupBy("y").agg(count($"marital")).show

df.groupBy("marital","y").count.show()

df.groupBy("age","y").count.show()

val df\_new = df.withColumn("age\_cat", when($"age" < 25 , "young").otherwise( when($"age" > 60 , "old").otherwise("mid\_age") ))

df\_new.groupBy("age\_cat","y").count.show()

**Output Screenshot:**

****