Report

On

Market Analysis in Banking Domain

With

Big Data Hadoop and Spark Developer

Business Scenario

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.

Domain: Banking (Market Analysis)

**Dataset Description**

 The data fields are as follows:

1. age numeric

2. job type of job (categorical: 'admin.','blue-collar','entrepreneur','housemaid','management','retired','self- employed', 'services','student','technician','unemployed','unknown')

3. marital marital status (categorical: 'divorced', 'married', 'single', 'unknown'; note: 'divorced' means divorced or widowed)

4. education (categorical: 'basic.4y','basic.6y','basic.9y','high.school','illiterate','professional.course','university.degree','unknown')

5. default has credit in default? (categorical: 'no', 'yes', 'unknown')

6. housing: has housing loan? (categorical: 'no', 'yes', 'unknown')

7. loan has a personal loan? (categorical: 'no', 'yes', 'unknown')

**# related to the last contact of the current campaign:**

8. contact contact communication type (categorical: 'cellular', 'telephone')

9. month Month of last contact (categorical: 'jan', 'feb', 'mar', ..., 'nov', 'dec')

10. day\_of\_week last contact day of the week (categorical: 'mon','tue','wed','thu','fri')

11. duration last contact duration, in seconds (numeric). Important note: this attribute highly affects the output target (example, if duration=0 then y='no'). Yet, the duration is not known before a call is performed. Also, after the end of the call “y” is obviously known. Thus, this input should only be included for benchmark purposes and should be discarded if the intention is to have a realistic predictive model.

**# other attributes:**

12. campaign number of times a customer was contacted during the campaign (numeric, includes last contact)

13. pdays: number of days passed after the customer was last contacted from a previous campaign (numeric; 999 means customer was not previously contacted)

14. previous number of times the customer was contacted prior to (or before) this campaign (numeric)

15. poutcome outcome of the previous marketing campaign (categorical: 'failure', 'nonexistent', 'success')

**#Output variable (desired target):**

16 y has the customer subscribed a term deposit? (binary: 'yes', 'no')

**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)
   1. Give marketing failure rate
3. Give the maximum, mean, and minimum age of the average targeted customer
4. Check the quality of customers by checking average balance, median balance of customers
5. Check if age matters in marketing subscription for deposit
6. Check if marital status mattered for a subscription to deposit
7. Check if age and marital status together mattered for a subscription to deposit scheme
8. Do feature engineering for the bank and find the right age effect on the campaign.

**Solution**

1. Load data and create a Spark data frame.

CODE:

**Step1:Read .csv File With sc.textFile(“Path”) and split rows with “;”**

val lines = sc.textFile("D:\\Project.csv")

val bank = lines.map(x => x.split(";"))

**Step2:Remove Header row which is column name with mapPartitionsWithIndex**

val bankf = bank.mapPartitionsWithIndex { (idx, iter) => if (idx == 0) iter.drop(1) else iter }

**Define Class for the schema**

case class Bank(age:Int, job:String, marital:String, education:String, defaultn:String, balance:Int,

housing:String, loan:String, contact:String, day:Int, month: String, duration:Int, campaign:Int, pdays:Int,

previous:Int, poutcome:String, y:String)

**Map Class Data to rdd and Create a TempView With toDF() Function.**

val bankrdd = bankf.map(

x => Bank(x(0).replaceAll("\"","").toInt,

x(1).replaceAll("\"","")

,x(2).replaceAll("\"","")

,x(3).replaceAll("\"","")

,x(4).replaceAll("\"","")

,x(5).replaceAll("\"","").toInt

,x(6).replaceAll("\"","")

,x(7).replaceAll("\"","")

,x(8).replaceAll("\"","")

,x(9).replaceAll("\"","").toInt

,x(10).replaceAll("\"","")

,x(11).replaceAll("\"","").toInt

,x(12).replaceAll("\"","").toInt

,x(13).replaceAll("\"","").toInt

,x(14).replaceAll("\"","").toInt

,x(15).replaceAll("\"","")

,x(16).replaceAll("\"","")

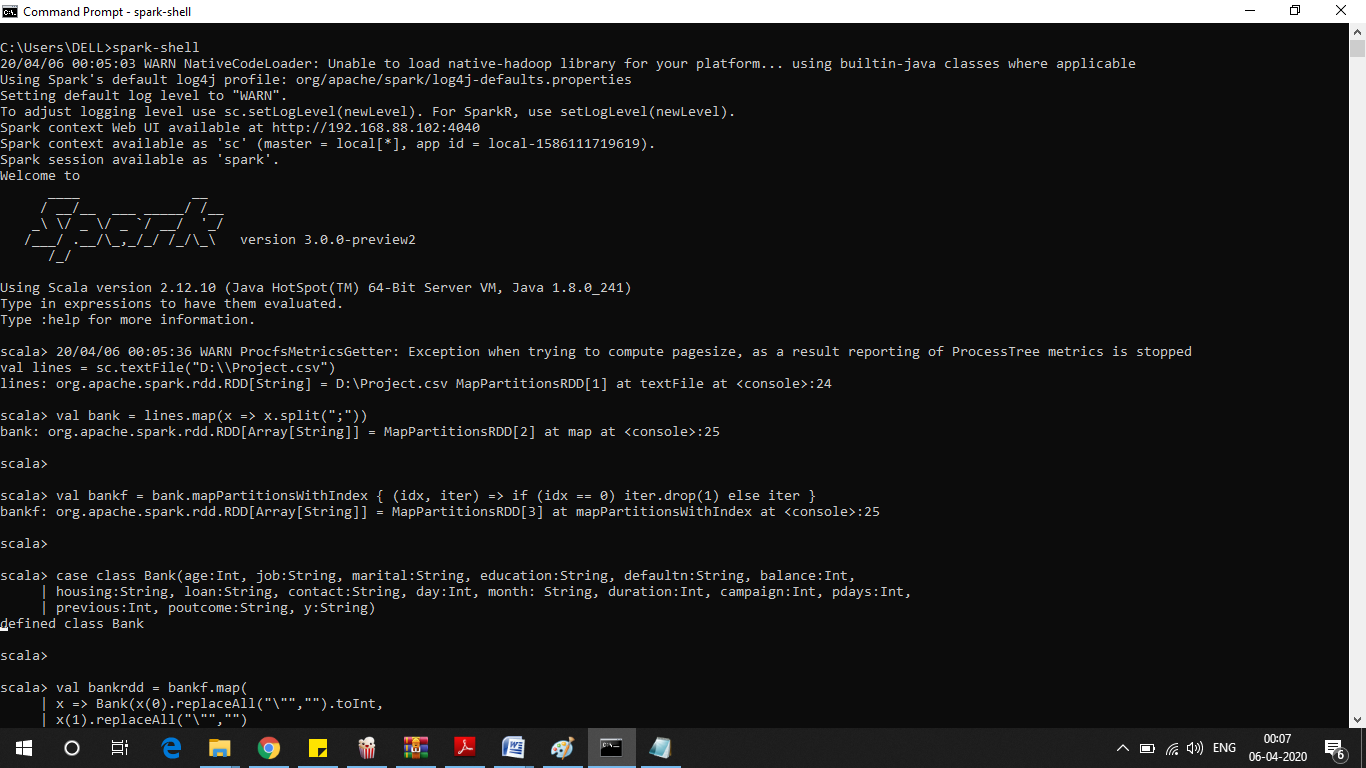
)

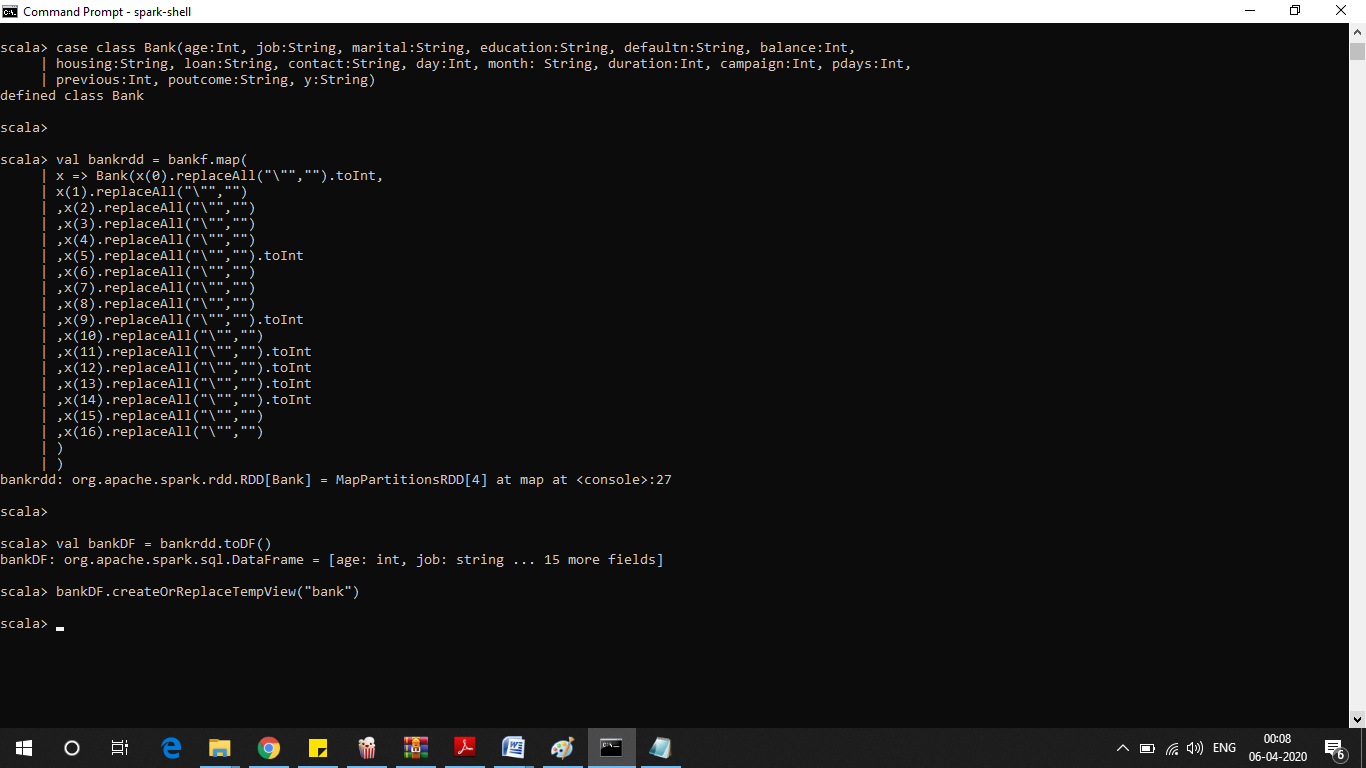
)

val bankDF = bankrdd.toDF()

bankDF.createOrReplaceTempView("bank")

Result:





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

CODE:

val ssb=new org.apache.spark.sql.SparkSession.Builder()

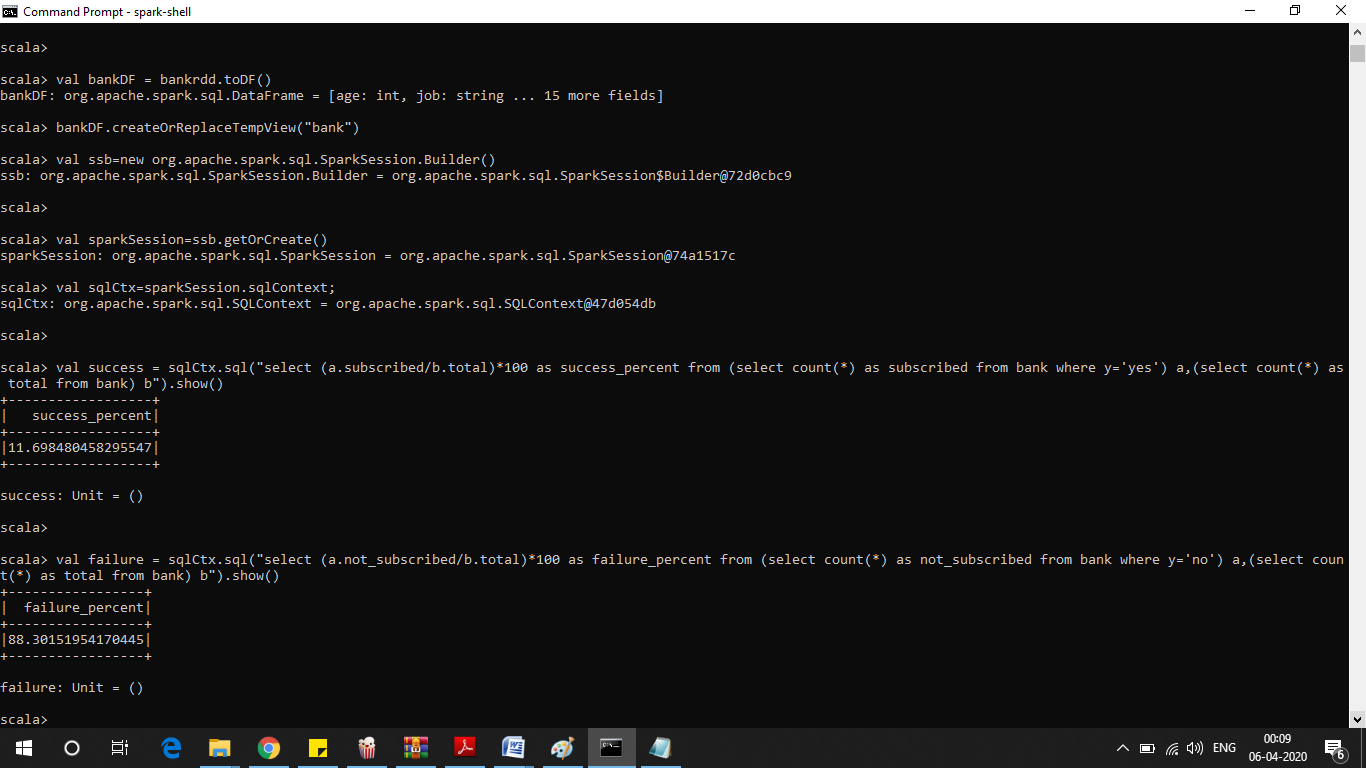
val sparkSession=ssb.getOrCreate()

val sqlCtx=sparkSession.sqlContext;

val success = sqlCtx.sql("select (a.subscribed/b.total)\*100 as success\_percent from (select count(\*) as subscribed from bank where y='yes') a,(select count(\*) as total from bank) b").show()

val failure = sqlCtx.sql("select (a.not\_subscribed/b.total)\*100 as failure\_percent from (select count(\*) as not\_subscribed from bank where y='no') a,(select count(\*) as total from bank) b").show()

RESULT:



.

1. Give the maximum, mean, and minimum age of the average targeted customer

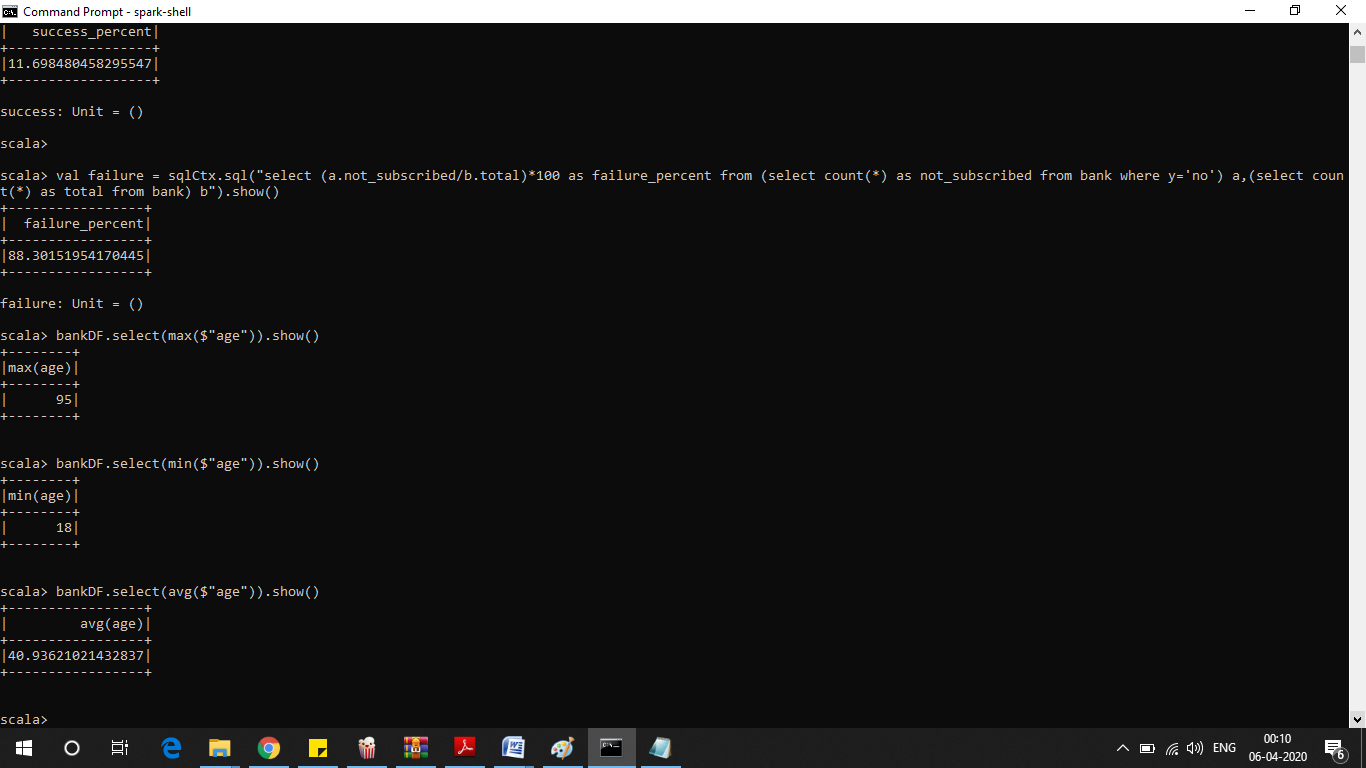
CODE:

bankDF.select(max($"age")).show()

bankDF.select(min($"age")).show()

bankDF.select(avg($"age")).show()

RESULT:



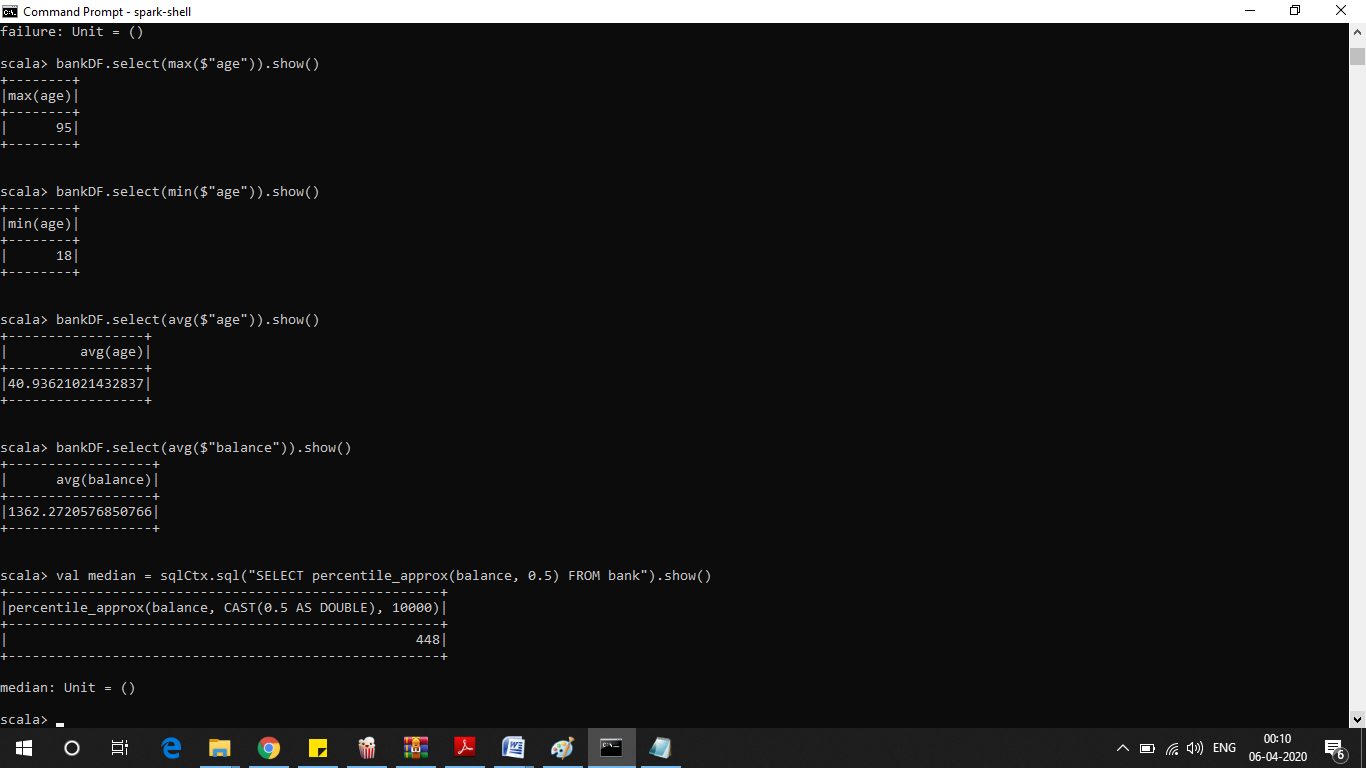
1. Check the quality of customers by checking average balance, median balance of customers

CODE:

bankDF.select(avg($"balance")).show()

val median = sqlCtx.sql("SELECT percentile\_approx(balance, 0.5) FROM bank").show()

RESULT:

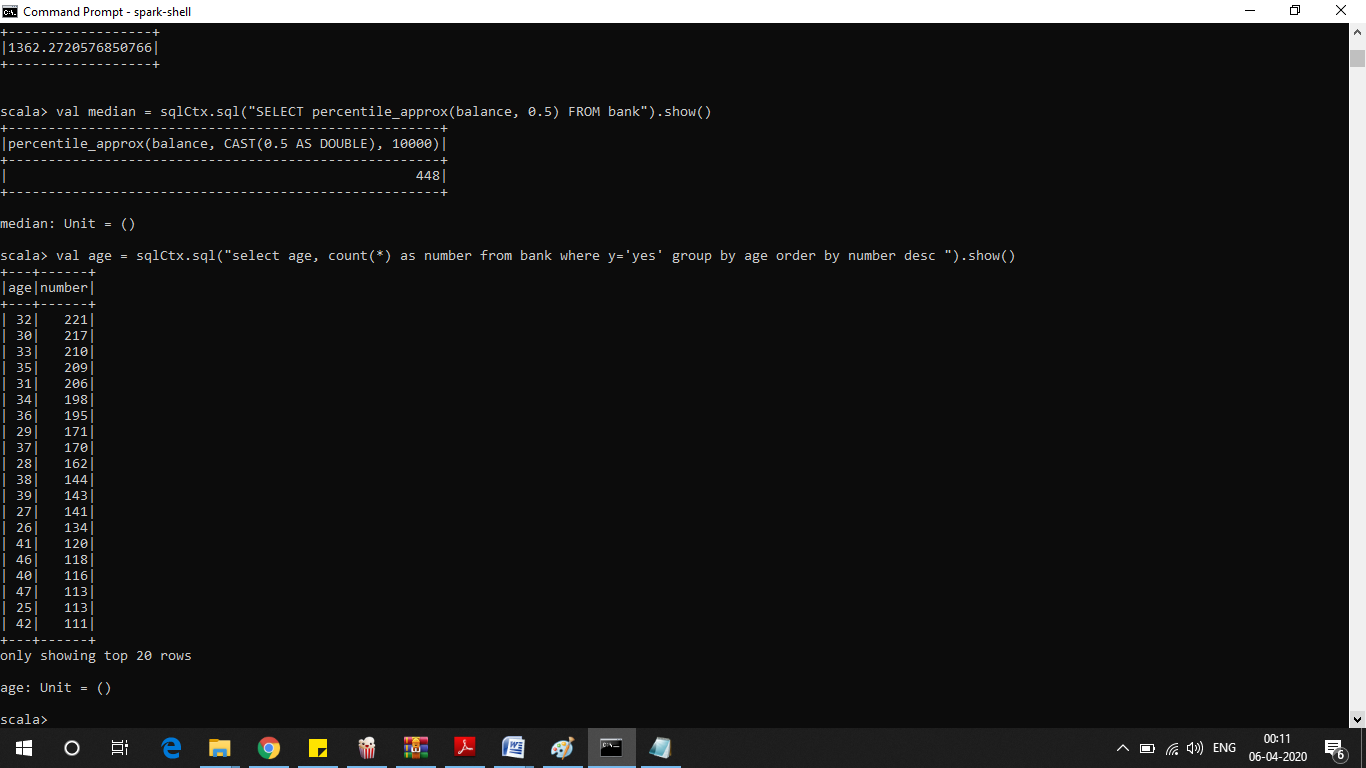


1. Check if age matters in marketing subscription for deposit

CODE:

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

RESULT:



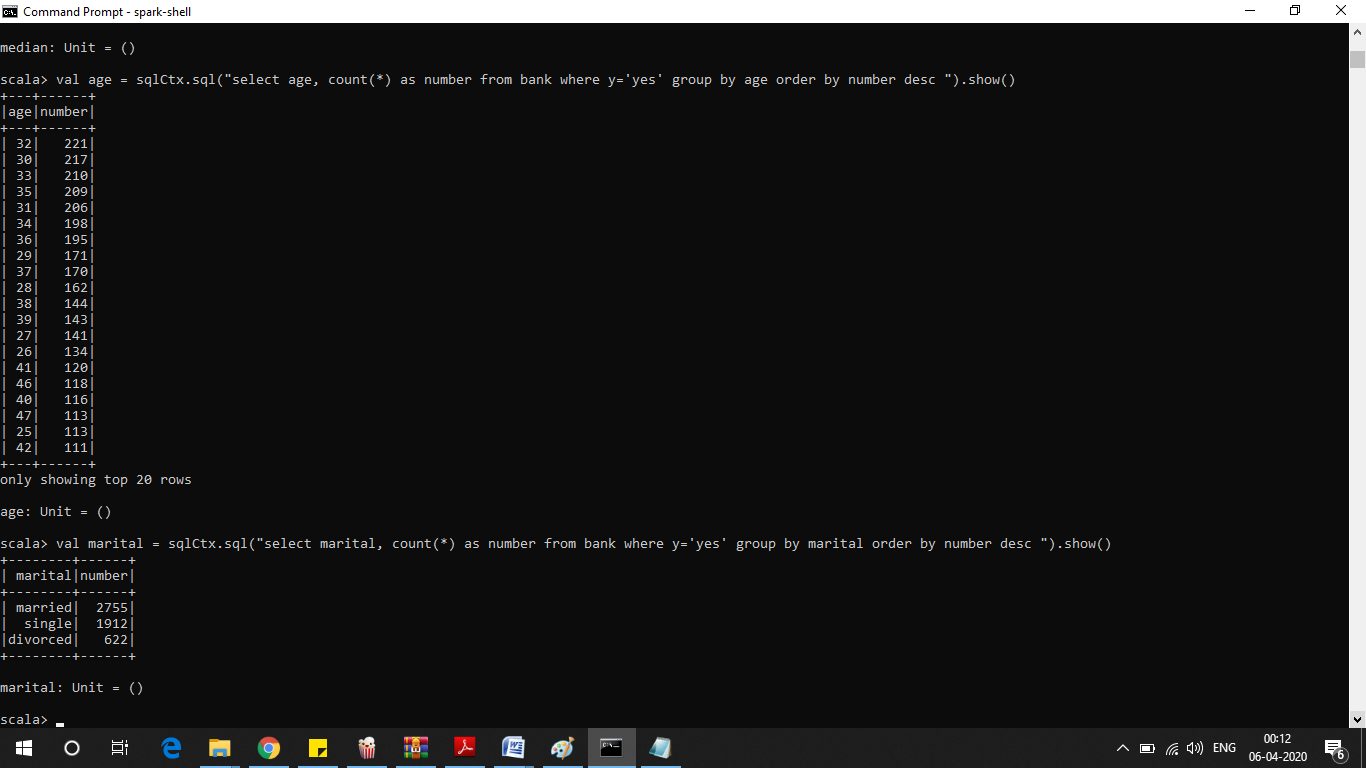
As above Result we see age indeed Matters The age Range 30 to 36 show most promises

1. Check if marital status mattered for a subscription to deposit

CODE:

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

RESULT:



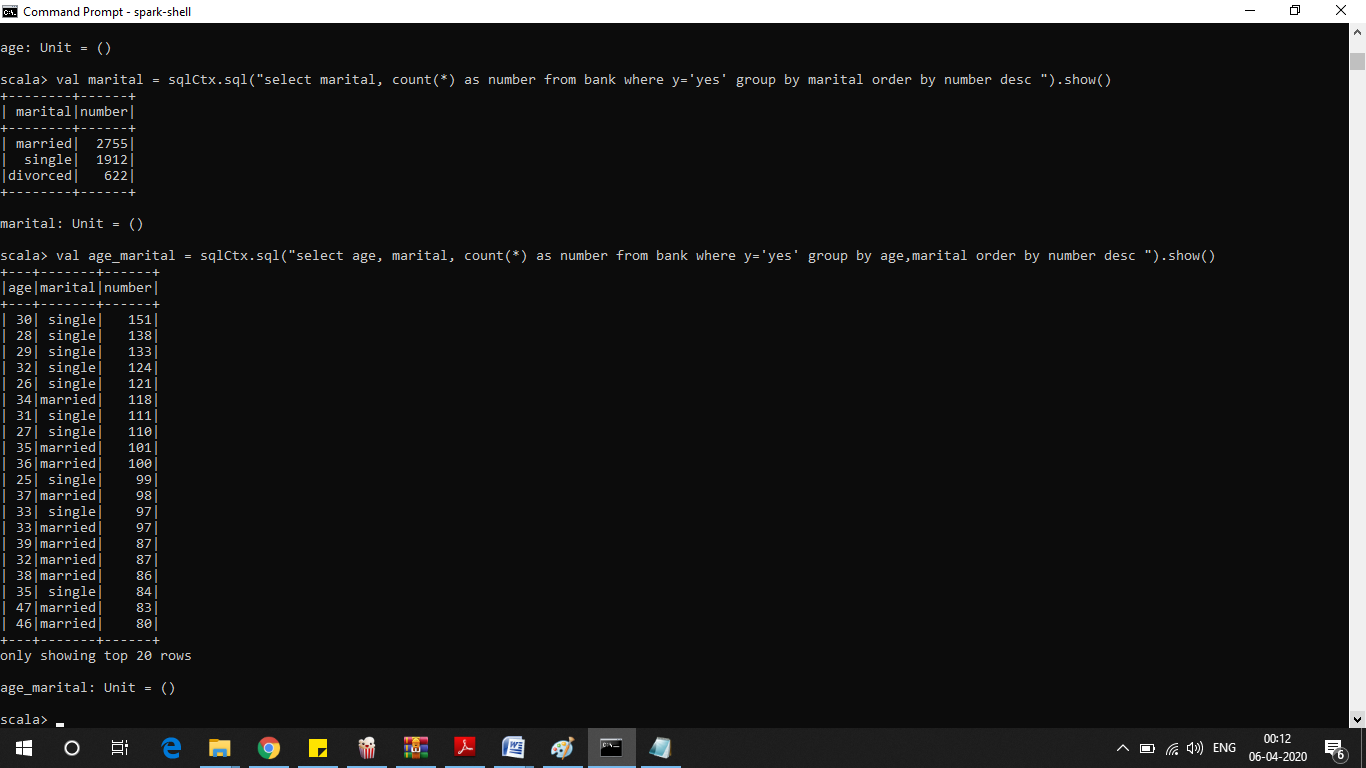
As above Result we see married couples who go for the subscriptions the most

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

CODE:

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

RESULT:



--Single people around the age 30-35 shows most subscriptions

1. Do feature engineering for the bank and find the right age effect on the campaign.

CODEL:

Import necessary libraries

import scala.reflect.runtime.universe

import org.apache.spark.SparkConf

import org.apache.spark.SparkContext

import org.apache.spark.sql.DataFrame

import org.apache.spark.sql.SQLContext

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

1--Defining a new UDF with which we will generate new features.We divide the age groups into 4

Categories

val ageRDD = sqlCtx.udf.register("ageRDD",(age:Int) => {

if (age < 20)

"Teen"

else if (age > 20 && age <= 32)

"Young"

else if (age > 33 && age <= 55)

"Middle Aged"

else

"Old"

})

2--Replacing old “age” column with new “age” column

val banknewDF = bankDF.withColumn("age",ageRDD(bankDF("age")))

banknewDF.createOrReplaceTempView("bank\_new")

Running a query to see the age group which subscribed the most

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

We see it’s ‘Middle-Aged’

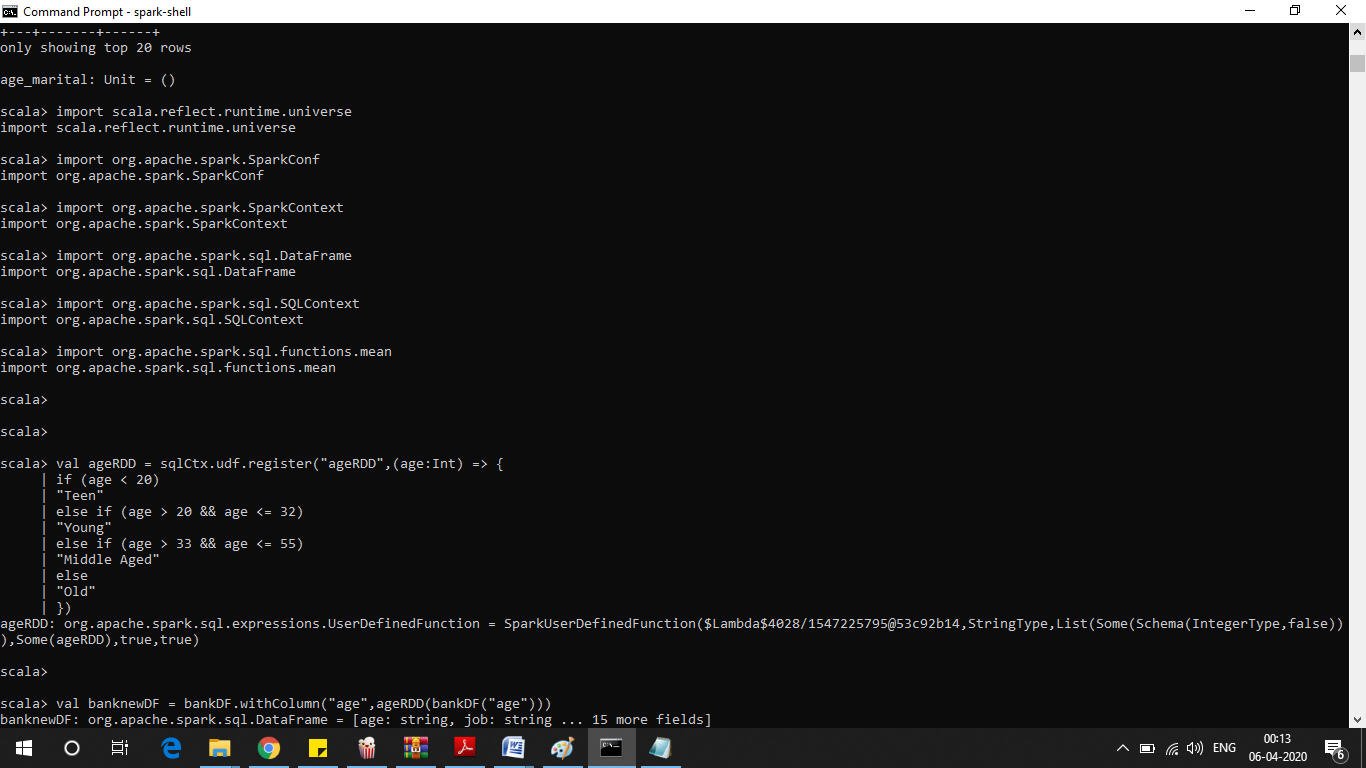
--Pipeline

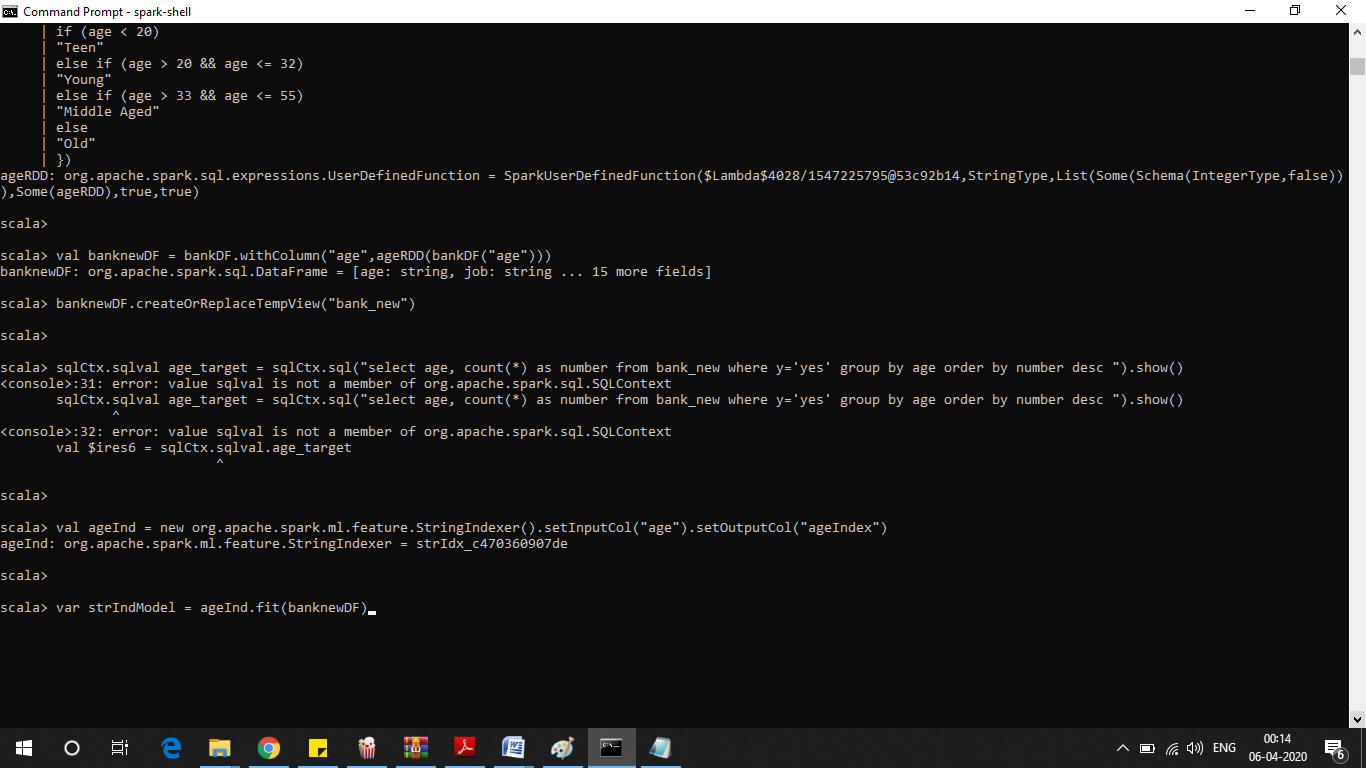
val ageInd = new org.apache.spark.ml.feature.StringIndexer().setInputCol("age").setOutputCol("ageIndex")

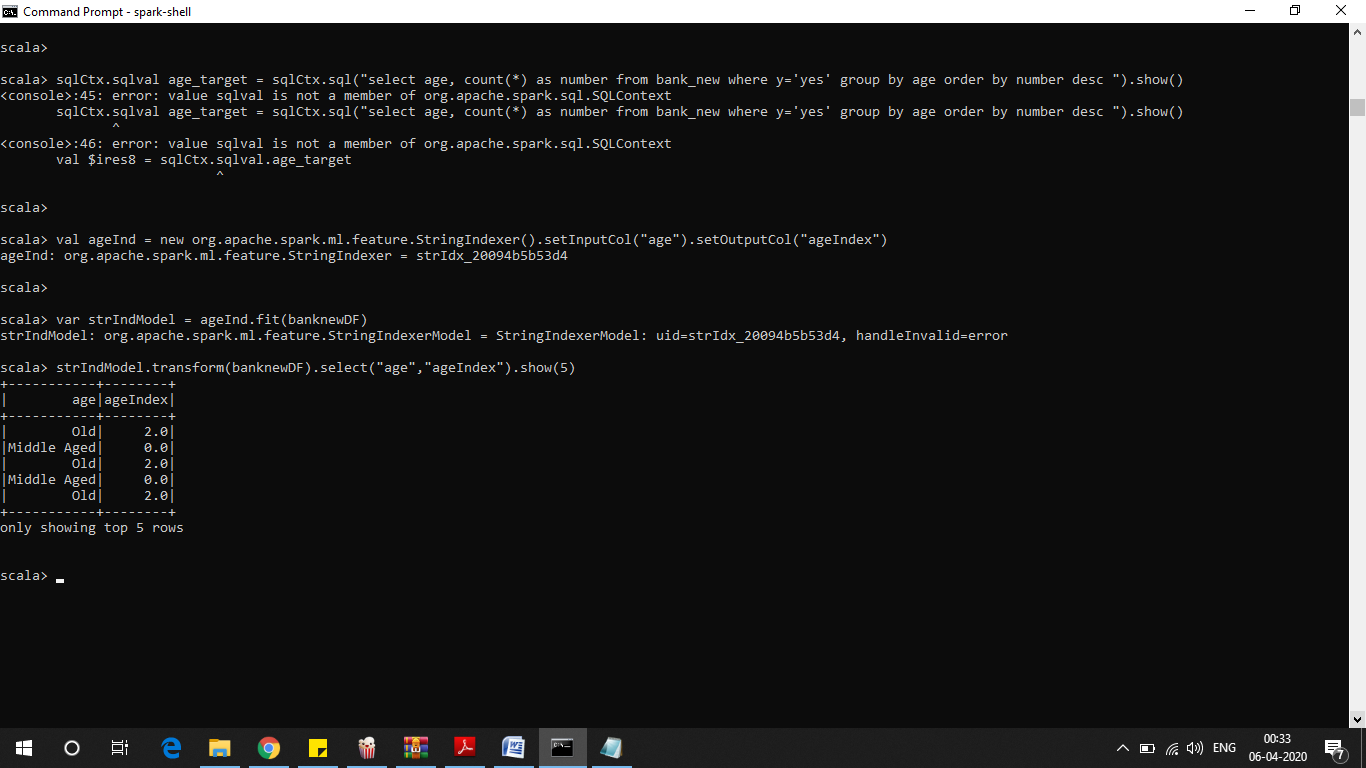
--Fitting the model

var strIndModel = ageInd.fit(banknewDF)

strIndModel.transform(banknewDF).select("age","ageIndex").show(5)







--Middle aged is the most frequent word in this data, so it is given index 0

So we can conclude from the Feature Engineering that It is the ‘Middle Aged’ people between age 33

and 55 who should be the targeted customers as they subscribe the most