1. Load data and create a Spark data frame

[ysamehayahoo@ip-10-0-31-28 ~]$ hdfs dfs -ls

Found 6 items

drwx------ - ysamehayahoo hadoop 0 2021-06-28 00:00 .Trash

drwxr-xr-x - ysamehayahoo hadoop 0 2021-06-29 18:44 .sparkStaging

drwxr-xr-x - ysamehayahoo hadoop 0 2021-06-29 19:07 Project1

drwxr-xr-x - ysamehayahoo hadoop 0 2021-06-28 11:21 Simplilearn

drwxr-xr-x - ysamehayahoo hadoop 0 2021-06-26 23:48 demo1

drwxr-xr-x - ysamehayahoo hadoop 0 2021-06-27 00:20 sam1

[ysamehayahoo@ip-10-0-31-28 ~]$ hdfs dfs -ls Project1

[ysamehayahoo@ip-10-0-31-28 ~]$ hdfs dfs -put project1.csv Project1

[ysamehayahoo@ip-10-0-31-28 ~]$ hdfs dfs -ls Project1

Found 1 items-rw-r--r-- 3 ysamehayahoo hadoop 5650234 2021-06-29 19:13 Project1/project1.csv

[ysamehayahoo@ip-10-0-31-28 ~]$ spark-shell

scala> val df = spark.read.option("header",true).csv("Project1/project1.csv")

df: org.apache.spark.sql.DataFrame = ["age;""job"";""marital"";""education"";""default"";""balance"";""housing"";""loan"";""contact"";""day"";""month"";""duration"";""campaign"";""pdays"";""previous"";""poutcome"";""y""": string]

scala> df.printSchema()

root

|--"age;""job"";""marital"";""education"";""default"";""balance"";""housing"";""loan"";""contact"";""day"";""month"";""duration"";""campaign"";""pdays"";""previous"";""poutcome"";""y""": string (nullable = true)

val txt = sc.textFile("Project1/project1.csv ")  
val bank = txt.map(x => x.split(";"))

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

case class Bank(age:Int, job:String, marital:String, education:String, defaulted: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)

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("\"","") ) )

**Text

Description automatically generated with medium confidence**

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

Give marketing failure rate

scala> val subscribed = sqlContext.sql("SELECT count(\*) as subscribed FROM bank Where y = 'yes'").show()

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

|subscribed|

+----------+|

|5289|

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

**Success-Rate**

println(5289.toFloat/total.toFloat\*100)

11.698221

scala> val unsubscribed = sqlContext.sql("SELECT count(\*) as subscribed FROM bank Where y = 'no'").show()

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

|subscribed|

+----------+|

|39922|

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

**Failure-Rate**

scala> println(39922.toFloat/total.toFloat\*100)

88.29957

**Graphical user interface, text, application

Description automatically generated**

3)Max,Min, Mean age of targeted customer

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

95

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

18

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

~41

Timeline

Description automatically generated with low confidence

4) Check the quality of customers by checking average balance, median balance of customers

Text

Description automatically generated

5) Check if age matters in marketing subscription for deposit

Text

Description automatically generated

Text

Description automatically generated6) Check if marital status mattered for a subscription to deposit

Text

Description automatically generated7) 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.

**Text

Description automatically generatedText

Description automatically generated**