Market Analysis in Banking Domain

(Source Code)

1. Load data and create a Spark data frame

ANS:

```
val mydf = spark.read.format("csv").
                option("header","true").
option("sep",";").

load("/user/gopipranay1997gmail/market_analysis_project/banking.csv")
mydf.printSchema
mydf.schema
mydf.show
mydf.count
```

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

ANS:

```
valsuc=mydf.filter($"y"==="yes").count.toFloat/mydf.count.toFloat*100
```

- Give marketing failure rate

ANS: val fail = mydf.filter($"y" === "no").count.toFloat /mydf.count.toFloat *100

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

ANS: import org.apache.spark.sql.functions.{min, max, avg}

mydf.agg(max($"age"),min($"age"), avg($"age")).show()

OR

import org.apache.commons.math3.stat.descriptive

mydf.createOrReplaceTempView("sample")

val med = sql("SELECT max(age) as max, min(age) as min, avg(age) as average, percentile_approx(age, 0.5) as median FROM sample");

med.show()

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

ANS: val medBal = sql("SELECT max(balance) as max, min(balance) as min, avg(balance) as average, percentile_approx(balance, 0.5) as median FROM sample");

medBal.show()

5. Check if age matters in marketing subscription for deposit

ANS: sql("select age,count(*) from banking where y='yes' group by age order by 2 desc").show()

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

ANS: sql("select marital,count(*) from banking where y='yes' group by marital order by 2 desc").show()

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

ANS: mydf.select("marital","age").filter('y==="yes").groupBy('marital,'age).count.sort(desc("count")).show

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

ANS: import org.apache.spark.sql.functions.udf

```
def ageToCategory = udf((age:Int) => {
    age match {
    case t if t < 25 => "young"
    case t if t > 60 => "Old"
    case _ => "mid"
    }
    }
   )

val newmydf
=mydf.withColumn("agecat",ageToCategory(mydf("age"))) //
create newcolumn
newmydf.groupBy("agecat","y").count().sort($"count".desc).sho
w
```