Market Analysis in Banking Domain

By

Ankita Das

1. Load data and create a Spark data frame

Import necessary Libraries

Import org.spark.sql.Dataframe

val bank_df = spark.read.option("header","true").option("inferSchema","true").csv("/user/dasankita07adgmail/Spark/bank-full.csv")

bank_df.show(5)
bank_df.printSchema()

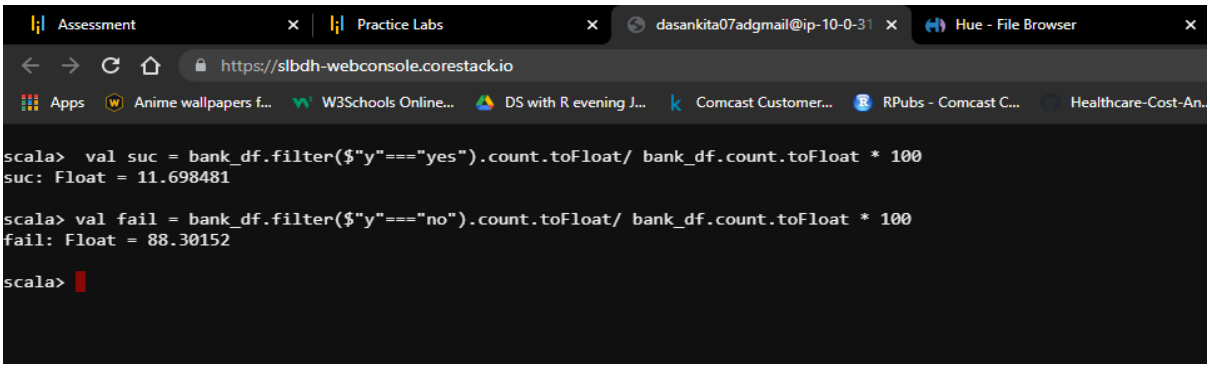
```
scala> bank_df.show(5)
+---+------------+-------+---------+-------+-------+-------+----+-------+---+-----+--------+--------+-----+--------+--------+---+
|age|         job|marital|education|default|balance|housing|loan|contact|day|month|duration|campaign|pdays|previous|poutcome|  y|
+---+------------+-------+---------+-------+-------+-------+----+-------+---+-----+--------+--------+-----+--------+--------+---+
| 58|  management|married| tertiary|     no|   2143|    yes|  no|unknown|  5|  may|     261|       1|   -1|       0| unknown| no|
| 44|  technician| single|secondary|     no|     29|    yes|  no|unknown|  5|  may|     151|       1|   -1|       0| unknown| no|
| 33|entrepreneur|married|secondary|     no|      2|    yes| yes|unknown|  5|  may|      76|       1|   -1|       0| unknown| no|
| 47| blue-collar|married|  unknown|     no|   1506|    yes|  no|unknown|  5|  may|      92|       1|   -1|       0| unknown| no|
| 33|     unknown| single|  unknown|     no|      1|     no|  no|unknown|  5|  may|     198|       1|   -1|       0| unknown| no|
+---+------------+-------+---------+-------+-------+-------+----+-------+---+-----+--------+--------+-----+--------+--------+---+
only showing top 5 rows


scala>  bank_df.printSchema()
root
 |-- age: integer (nullable = true)
 |-- job: string (nullable = true)
 |-- marital: string (nullable = true)
 |-- education: string (nullable = true)
 |-- default: string (nullable = true)
 |-- balance: integer (nullable = true)
 |-- housing: string (nullable = true)
 |-- loan: string (nullable = true)
 |-- contact: string (nullable = true)
 |-- day: integer (nullable = true)
 |-- month: string (nullable = true)
 |-- duration: integer (nullable = true)
 |-- campaign: integer (nullable = true)
 |-- pdays: integer (nullable = true)
 |-- previous: integer (nullable = true)
 |-- poutcome: string (nullable = true)
 |-- y: string (nullable = true)
```

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

val suc = bank_df.filter($"y"==="yes").count.toFloat/ bank_df.count.toFloat * 100

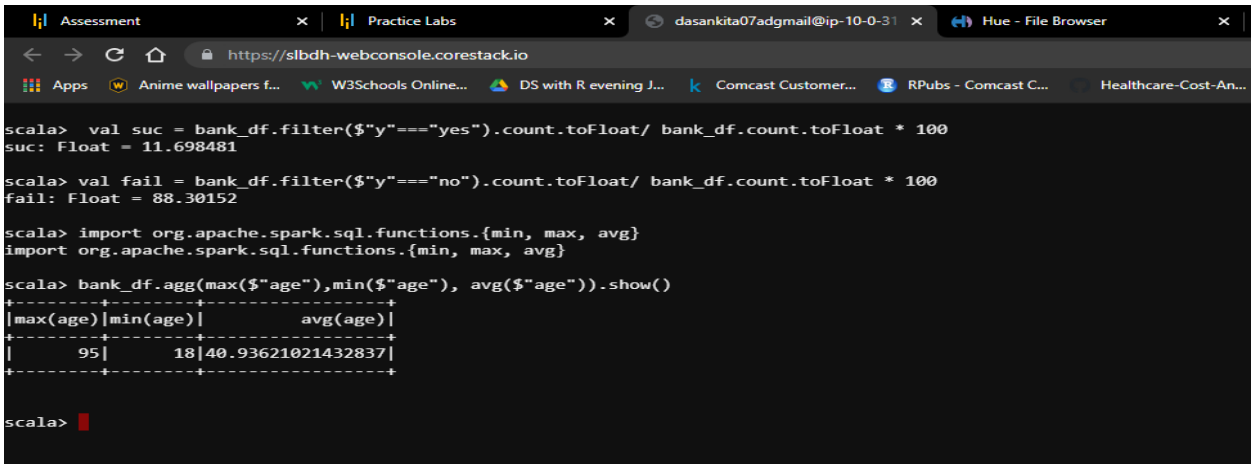
val fail = bank_df.filter($"y"==="no").count.toFloat/ bank_df.count.toFloat * 100



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

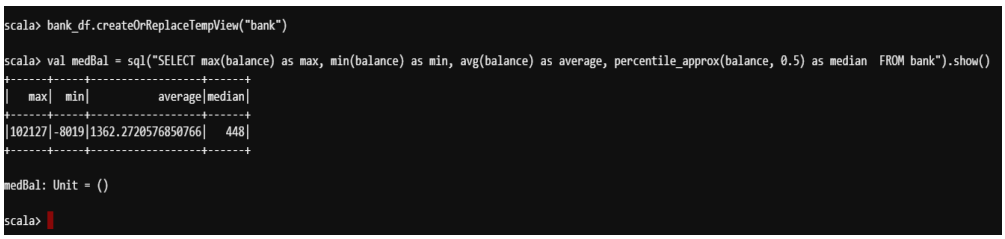
```
import org.apache.spark.sql.functions.{min, max, avg}

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



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

```
bank_df.createOrReplaceTempView("bank")
val medBal = sql("SELECT max(balance) as max, min(balance) as min, avg(balance) as average,
percentile_approx(balance, 0.5) as median  FROM bank").show()
```



5. Check if age matters in marketing subscription for deposit

bank_df.filter($"y"==="yes").groupBy("age").count().orderBy("age").show(25)

```
scala>  bank_df.filter($"y"==="yes").groupBy("age").count().orderBy("age").show(25)
+---+-----+
|age|count|
+---+-----+
| 18|    7|
| 19|   11|
| 20|   15|
| 21|   22|
| 22|   40|
| 23|   44|
| 24|   68|
| 25|  113|
| 26|  134|
| 27|  141|
| 28|  162|
| 29|  171|
| 30|  217|
| 31|  206|
| 32|  221|
| 33|  210|
| 34|  198|
| 35|  209|
| 36|  195|
| 37|  170|
| 38|  144|
| 39|  143|
| 40|  116|
| 41|  120|
| 42|  111|
+---+-----+
only showing top 25 rows
```

As in the result , we Can see that age between 28-40 , has subscribed more than other age peoples. So , we can conclude that age matters.

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

bank_df.filter($"y"==="yes").groupBy("marital").count().orderBy("marital").show()

```
scala>  bank_df.filter($"y"==="yes").groupBy("marital").count().orderBy("marital").show()
+--------+-----+
| marital|count|
+--------+-----+
|divorced|  622|
| married| 2755|
|  single| 1912|
+--------+-----+
```

As in result we can see that Marriage people has subscribed more than Single & divorced., So marital status also matters

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

As age is continuous data, for filter it we can create a user defined functions to make a new column and categorized it (it is a part of feature engineering)

```
import org.apache.spark.sql.functions.udf

def age_cat = udf((age:Int) => {
    | if (age < 19)
    | "Teen"
    | else if(age > 19 && age <= 30)
    | "Young"
    | else if(age > 30 && age <= 55)
    | "Mid Age"
    | else
    | "Old"
    | })

val bank_new_df = bank_df.withColumn("agecat",age_cat(bank_df("age")))

bank_new_df.filter($"y"==="yes").groupBy("marital","agecat").count().orderBy("marital").show()
```

```
scala> bank_new_df.filter($"y"==="yes").groupBy("marital","agecat").count().orderBy("marital").show()
+--------+-------+-----+
| marital| agecat|count|
+--------+-------+-----+
|divorced|    Old|  179|
|divorced|Mid Age|  425|
|divorced|  Young|   18|
| married|  Young|  182|
| married|Mid Age| 1877|
| married|    Old|  696|
|  single|  Young|  927|
|  single|    Old|   31|
|  single|Mid Age|  936|
|  single|   Teen|   18|
+--------+-------+-----+
```

As in result, We can conclude that Married mid age peoples are more likely to subscribed term deposite

6. Find the right age effect on the campaign.

```
bank_new_df.filter($”y”==”yes”).groupBy(“agecat”).count().orderBy(“agecat”).show()
```

```
scala> bank_new_df.filter($"y"==="yes").groupBy("agecat").count().orderBy("agecat").show()
+-------+-----+
| agecat|count|
+-------+-----+
|Mid Age| 3238|
|    Old|  906|
|   Teen|   18|
|  Young| 1127|
+-------+-----+
```

As we can conclude that Mid Age peoples are more like to subscribed.