Project 3: Market Analysis in Banking Domain

DESCRIPTION

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:

Index	Variables	Description
1.	age	numeric
2.	job	type of job (categorical: 'admin.','blue-collar','entrepreneur','housemaid','management','retired','s employed','services','student','technician','unemployed','unknown')
3.	marital	marital status (categorical: 'divorced', 'married', 'single', 'unknown'; note: 'divorced' means divor widowed)
4.	education	(categorical: 'basic.4y','basic.6y','basic.9y','high.school','illiterate','professional.course','university.degree','unk
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:

Index	Variables	Description
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:

Index	Variables	Description
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):

Index	Variables	Description
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)
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.

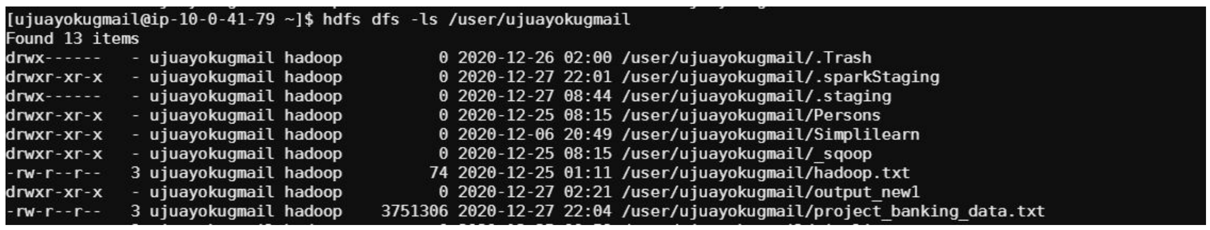
Note:

The project was executed with the provided labs: WebConsole, FTP, Spark, Hadoop etc. The steps taken and screenshots of the results are compiled as shown below

1. Load data and create a Spark data frame

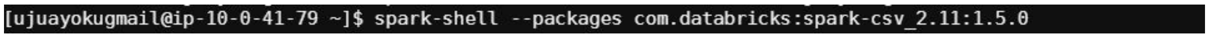
Uploaded the dataset file to HDFS via FTP.

Checking to verify that the upload was successful:



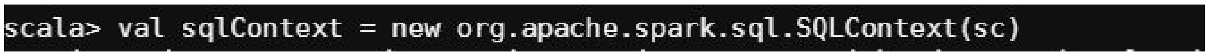
```
[ujuayokugmail@ip-10-0-41-79 ~]$ hdfs dfs -ls /user/ujuayokugmail
Found 13 items
drwx------   - ujuayokugmail hadoop          0 2020-12-26 02:00 /user/ujuayokugmail/.Trash
drwxr-xr-x   - ujuayokugmail hadoop          0 2020-12-27 22:01 /user/ujuayokugmail/.sparkStaging
drwx------   - ujuayokugmail hadoop          0 2020-12-27 08:44 /user/ujuayokugmail/.staging
drwxr-xr-x   - ujuayokugmail hadoop          0 2020-12-25 08:15 /user/ujuayokugmail/Persons
drwxr-xr-x   - ujuayokugmail hadoop          0 2020-12-06 20:49 /user/ujuayokugmail/Simplilearn
drwxr-xr-x   - ujuayokugmail hadoop          0 2020-12-25 08:15 /user/ujuayokugmail/_sqoop
-rw-r--r--   3 ujuayokugmail hadoop         74 2020-12-25 01:11 /user/ujuayokugmail/hadoop.txt
drwxr-xr-x   - ujuayokugmail hadoop          0 2020-12-27 02:21 /user/ujuayokugmail/output_new1
-rw-r--r--   3 ujuayokugmail hadoop    3751306 2020-12-27 22:04 /user/ujuayokugmail/project_banking_data.txt
```

Run spark shell:



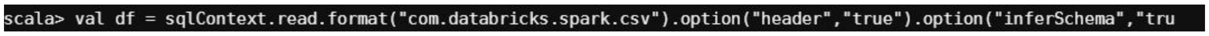
```
[ujuayokugmail@ip-10-0-41-79 ~]$ spark-shell --packages com.databricks:spark-csv_2.11:1.5.0
```

Create SQL context:



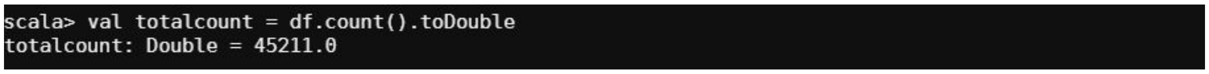
```
scala> val sqlContext = new org.apache.spark.sql.SQLContext(sc)
```

Load the file as dataframe using the spark-csv package:



```
scala> val df = sqlContext.read.format("com.databricks.spark.csv").option("header","true").option("inferSchema","tru
```

Get the total count of records:

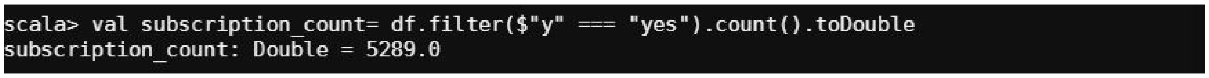


```
scala> val totalcount = df.count().toDouble
totalcount: Double = 45211.0
```

2. Give marketing success rate

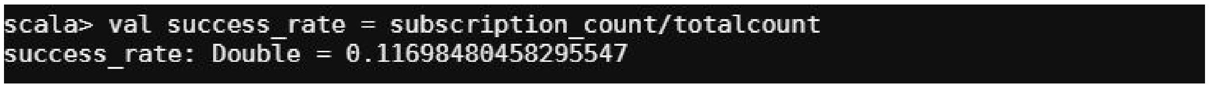
Success rate = No. of people subscribed / total no. of entries

Get the total number of subscribed clients:



```
scala> val subscription_count= df.filter($"y" === "yes").count().toDouble
subscription_count: Double = 5289.0
```

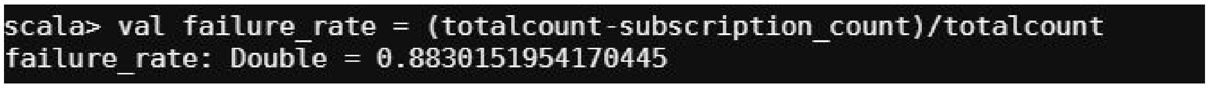
Calculating the success rate using the formula provided:



```
scala> val success_rate = subscription_count/totalcount
success_rate: Double = 0.11698480458295547
```

The marketing success rate is **11.7%**

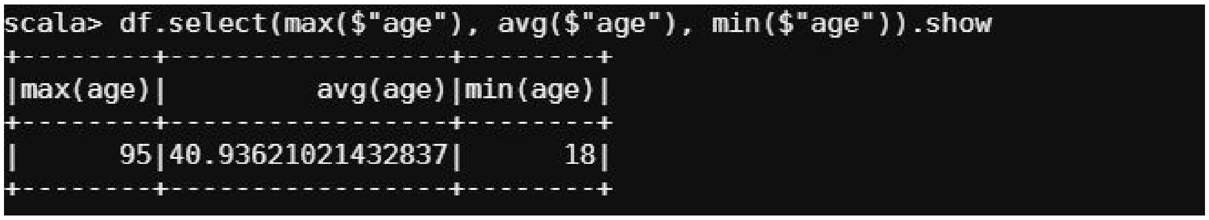
3. Give marketing failure rate



```
scala> val failure_rate = (totalcount-subscription_count)/totalcount
failure_rate: Double = 0.8830151954170445
```

The marketing failure rate is **88.3%**

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



```
scala> df.select(max($"age"), avg($"age"), min($"age")).show
+--------+----------------+--------+
|max(age)|        avg(age)|min(age)|
+--------+----------------+--------+
|      95|40.93621021432837|      18|
+--------+----------------+--------+
```

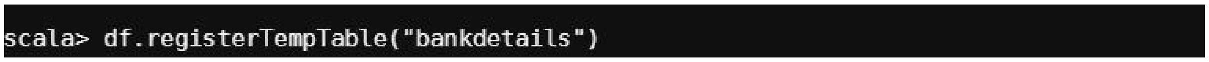
Maximum age: 95 years

Average age: 40years

Min Age: 18 years

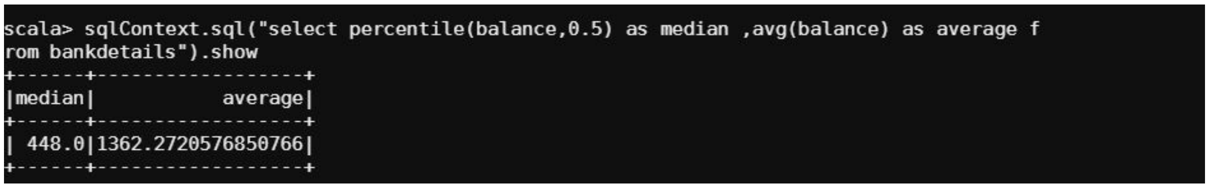
5. Check the quality of customers by checking average balance, median balance of customers

Register the temp table with the name "bankdetails":



```
scala> df.registerTempTable("bankdetails")
```

Get the average and median balance from the table:

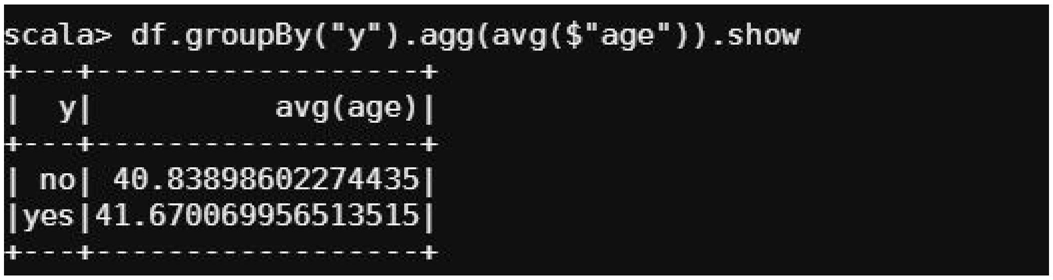


```
scala> sqlContext.sql("select percentile(balance,0.5) as median ,avg(balance) as average f
rom bankdetails").show
+------+------------------+
|median|           average|
+------+------------------+
| 448.0|1362.2720576850766|
+------+------------------+
```

Median Balance: 448.0 Average Balance: 1362.27

6. Check if age matters in marketing subscription for deposit

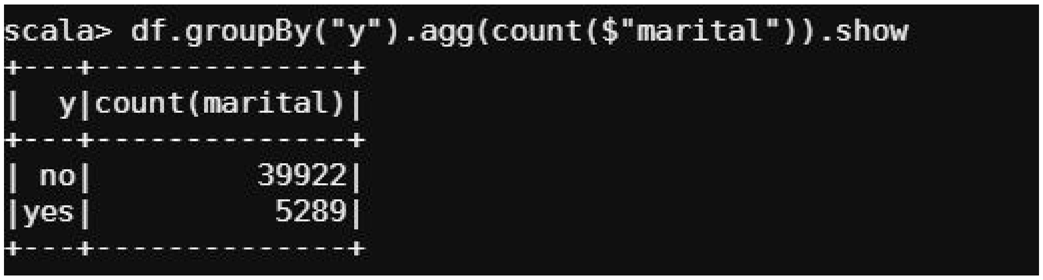
Get the average age based on subscriptions:



```
scala> df.groupBy("y").agg(avg($"age")).show
+---+-----------------+
|  y|         avg(age)|
+---+-----------------+
| no| 40.83898602274435|
|yes|41.670069956513515|
+---+-----------------+
```

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

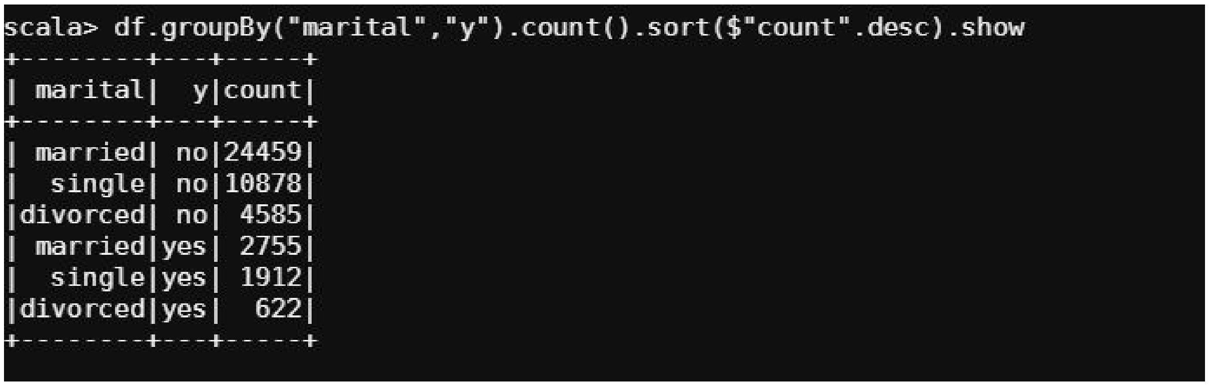
Get the count based on the subscription on the marital status:



```
scala> df.groupBy("y").agg(count($"marital")).show
+---+--------------+
|  y|count(marital)|
+---+--------------+
| no|         39922|
|yes|          5289|
+---+--------------+
```

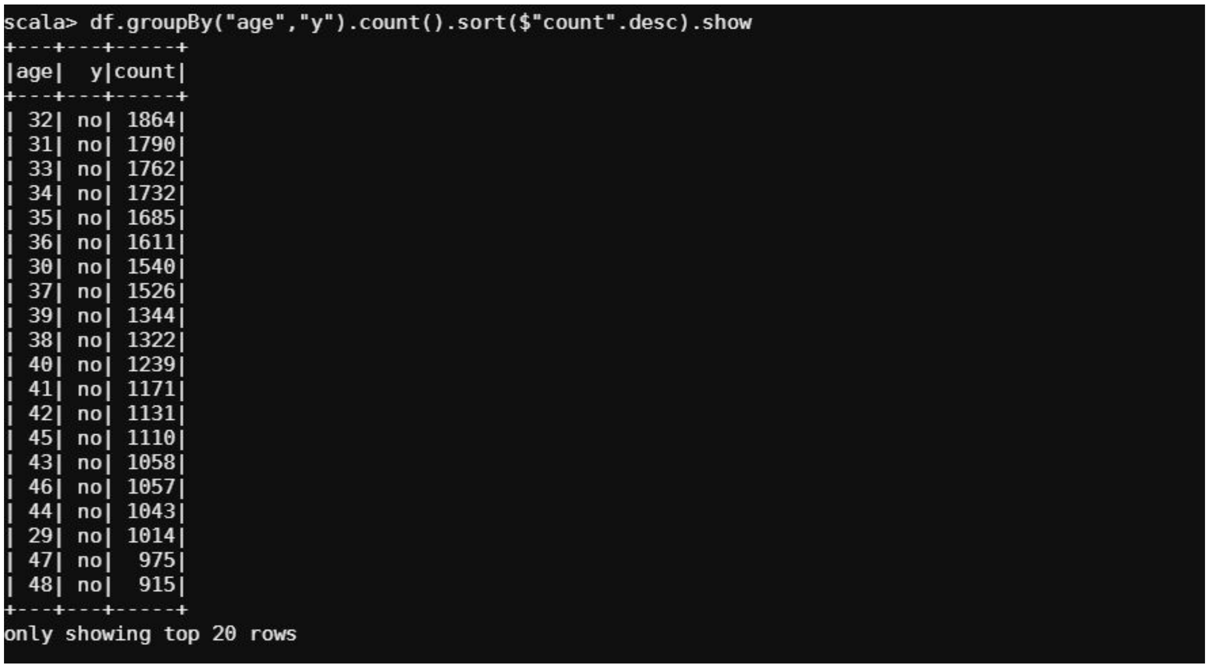
8. Check if age and marital status together mattered for a subscription to deposit scheme

Get the grouping based on age and the marital status:

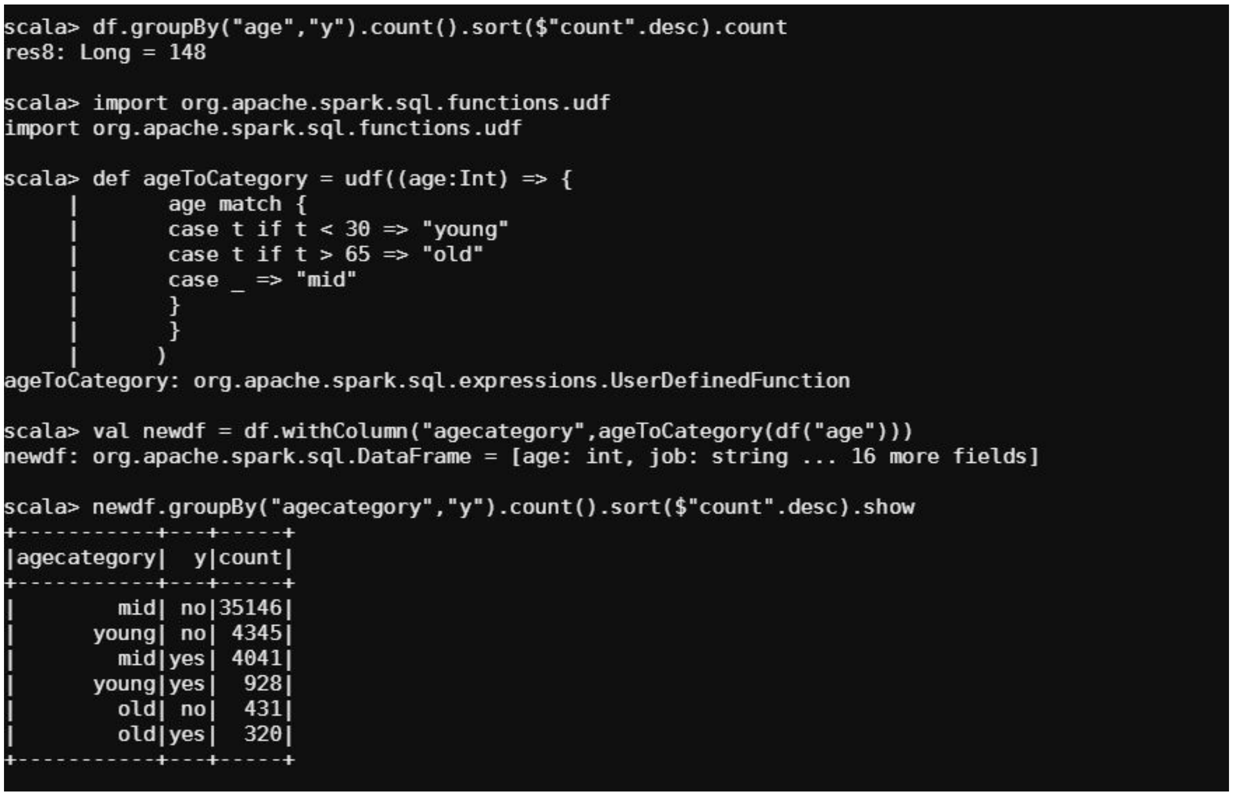


```
scala> df.groupBy("marital","y").count().sort($"count".desc).show
+--------+---+-----+
| marital|  y|count|
+--------+---+-----+
| married| no|24459|
|  single| no|10878|
|divorced| no| 4585|
| married|yes| 2755|
|  single|yes| 1912|
|divorced|yes|  622|
+--------+---+-----+
```

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



```
scala> df.groupBy("age","y").count().sort($"count".desc).show
+---+---+-----+
|age|  y|count|
+---+---+-----+
| 32| no| 1864|
| 31| no| 1790|
| 33| no| 1762|
| 34| no| 1732|
| 35| no| 1685|
| 36| no| 1611|
| 30| no| 1540|
| 37| no| 1526|
| 39| no| 1344|
| 38| no| 1322|
| 40| no| 1239|
| 41| no| 1171|
| 42| no| 1131|
| 45| no| 1110|
| 43| no| 1058|
| 46| no| 1057|
| 44| no| 1043|
| 29| no| 1014|
| 47| no|  975|
| 48| no|  915|
+---+---+-----+
only showing top 20 rows
```



```
scala> df.groupBy("age","y").count().sort($"count".desc).count
res8: Long = 148

scala> import org.apache.spark.sql.functions.udf
import org.apache.spark.sql.functions.udf

scala> def ageToCategory = udf((age:Int) => {
     |       age match {
     |       case t if t < 30 => "young"
     |       case t if t > 65 => "old"
     |       case _ => "mid"
     |       }
     |       }
     |      )
ageToCategory: org.apache.spark.sql.expressions.UserDefinedFunction

scala> val newdf = df.withColumn("agecategory",ageToCategory(df("age")))
newdf: org.apache.spark.sql.DataFrame = [age: int, job: string ... 16 more fields]

scala> newdf.groupBy("agecategory","y").count().sort($"count".desc).show
+-----------+---+-----+
|agecategory|  y|count|
+-----------+---+-----+
|        mid| no|35146|
|      young| no| 4345|
|        mid|yes| 4041|
|      young|yes|  928|
|        old| no|  431|
|        old|yes|  320|
+-----------+---+-----+
```

Middle aged are more likely to subscribe than other age groups.