

PROJECT 3:

Market Analysis in Banking Domain

AUTHOR: Manasa Devikoppa Basavarajappa

SUBMISSION DATE: 26 November, 2020

TABLE OF INDEX

CHAPTER 1: PROBLEM STATEMENT	3
CHAPTER 2: WRITEUP	5
CHAPTER 3: SOURCE CODE WITH OUTPUT	6

LIST OF FIGURES

No table of figures entries found.

CHAPTER 1: PROBLEM STATEMENT

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).
 - 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.

CHAPTER 2: WRITEUP

The project is related to the marketing analysis of the Portuguese Banking Institution. The goal of the project is to perform the analysis of the data generated by the marketing campaign.

The 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.

The project uses Scala as a programming language, Hive as a data storage unit and Spark SQL functions to achieve the results.

CHAPTER 3: SOURCE CODE WITH OUTPUT

Task 1: Load data and create a Spark data frame.

```
scala> import org.apache.spark.sql.SQLContext
```

```
scala> val sqlContext = new SQLContext(sc)
```

```
scala> val df = sqlContext.read.format("com.databricks.spark.csv").option("header",
"true").option("inferSchema", "true").load("/user/manasadbgmail/Spark_Project_Marketing_
Analytics/Input_Data/Marketing_Analysis.csv")
```

```
scala> df.show
```

Output 1:

```
warning: there was one deprecation warning; re-run with -deprecation for details
sqlContext: org.apache.spark.sql.SQLContext = org.apache.spark.sql.SQLContext@2c8d537e

scala> val df = sqlContext.read.format("com.databricks.spark.csv").option("header", "true").option("inferSchema", "true").load("/user/manasadbgmail/Spark_Project_Marketing_Analytics/Input_Data/Marketing_Analysis.csv")
df: org.apache.spark.sql.DataFrame = [age: int, job: string ... 15 more fields]

scala> df.show
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|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|
| 35| management| married| tertiary|    no|   231|    yes|   no| unknown|  5|   may|    139|        1|    -1|        0| unknown| no|
| 28| management| single| tertiary|    no|   447|    yes| yes| unknown|  5|   may|    217|        1|    -1|        0| unknown| no|
| 42| entrepreneur| divorced| tertiary|   yes|     2|    yes|   no| unknown|  5|   may|    380|        1|    -1|        0| unknown| no|
| 58|    retired| married| primary|    no|   121|    yes|   no| unknown|  5|   may|     50|        1|    -1|        0| unknown| no|
| 43| technician| single| secondary|    no|   593|    yes|   no| unknown|  5|   may|     55|        1|    -1|        0| unknown| no|
| 41|    admin.| divorced| secondary|    no|   270|    yes|   no| unknown|  5|   may|    222|        1|    -1|        0| unknown| no|
| 29|    admin.| single| secondary|    no|   390|    yes|   no| unknown|  5|   may|    137|        1|    -1|        0| unknown| no|
| 53| technician| married| secondary|    no|     6|    yes|   no| unknown|  5|   may|    517|        1|    -1|        0| unknown| no|
| 58| technician| married| unknown|    no|    71|    yes|   no| unknown|  5|   may|     71|        1|    -1|        0| unknown| no|
| 57|    services| married| secondary|    no|   162|    yes|   no| unknown|  5|   may|    174|        1|    -1|        0| unknown| no|
| 51|    retired| married| primary|    no|   229|    yes|   no| unknown|  5|   may|    353|        1|    -1|        0| unknown| no|
| 45|    admin.| single| unknown|    no|    13|    yes|   no| unknown|  5|   may|     98|        1|    -1|        0| unknown| no|
| 57| blue-collar| married| primary|    no|    52|    yes|   no| unknown|  5|   may|     38|        1|    -1|        0| unknown| no|
| 60|    retired| married| primary|    no|    60|    yes|   no| unknown|  5|   may|    219|        1|    -1|        0| unknown| no|
| 33|    services| married| secondary|    no|     0|    yes|   no| unknown|  5|   may|     54|        1|    -1|        0| unknown| no|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
only showing top 20 rows

scala>
```

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

```
scala> val tot_count=df.count()

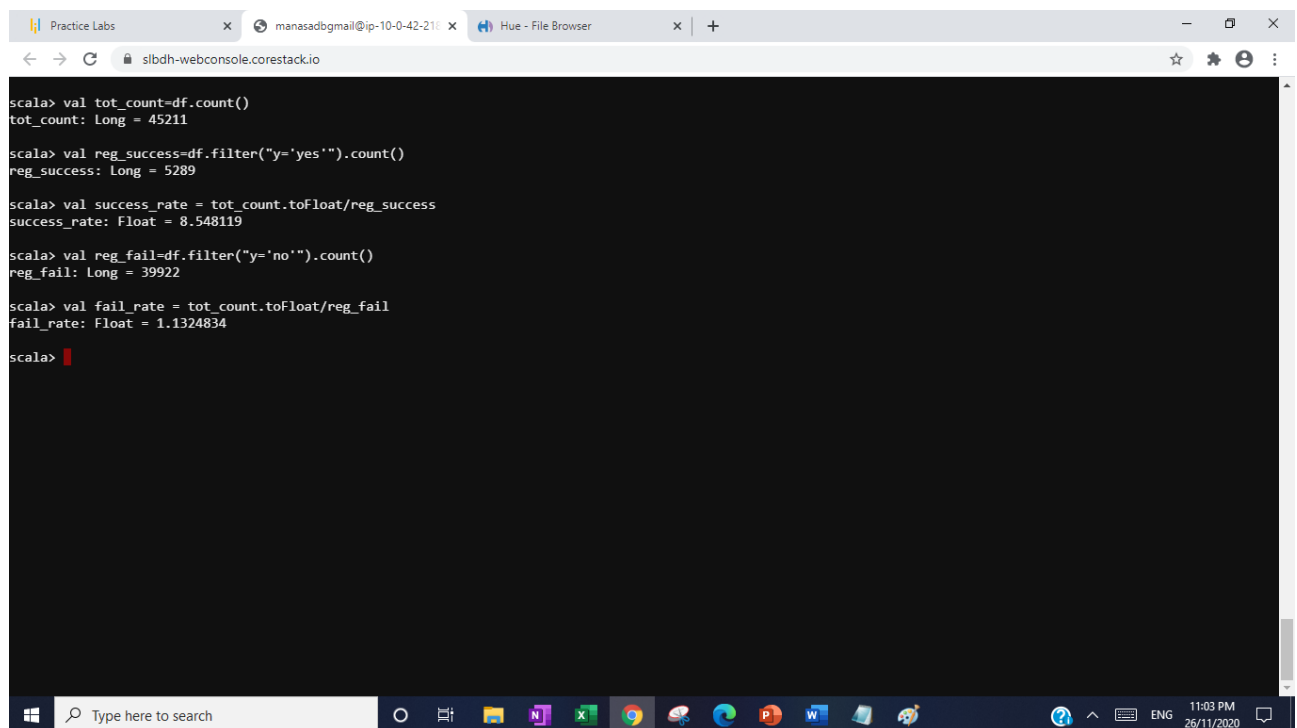
scala> val reg_success=df.filter("y='yes']").count()

scala> val success_rate = tot_count.toFloat/reg_success

scala> val reg_fail=df.filter("y='no']").count()

scala> val fail_rate = tot_count.toFloat/reg_fail
```

Output 2:

A screenshot of a web browser window displaying a Scala REPL interface. The browser has three tabs: 'Practice Labs', 'manasadbgmail@ip-10-0-42-210', and 'Hue - File Browser'. The address bar shows 'slbdh-webconsole.corestack.io'. The console output shows the following Scala code and results:

```
scala> val tot_count=df.count()
tot_count: Long = 45211

scala> val reg_success=df.filter("y='yes']").count()
reg_success: Long = 5289

scala> val success_rate = tot_count.toFloat/reg_success
success_rate: Float = 8.548119

scala> val reg_fail=df.filter("y='no']").count()
reg_fail: Long = 39922

scala> val fail_rate = tot_count.toFloat/reg_fail
fail_rate: Float = 1.1324834

scala>
```

The Windows taskbar is visible at the bottom, showing the search bar and various application icons. The system clock indicates 11:03 PM on 26/11/2020.

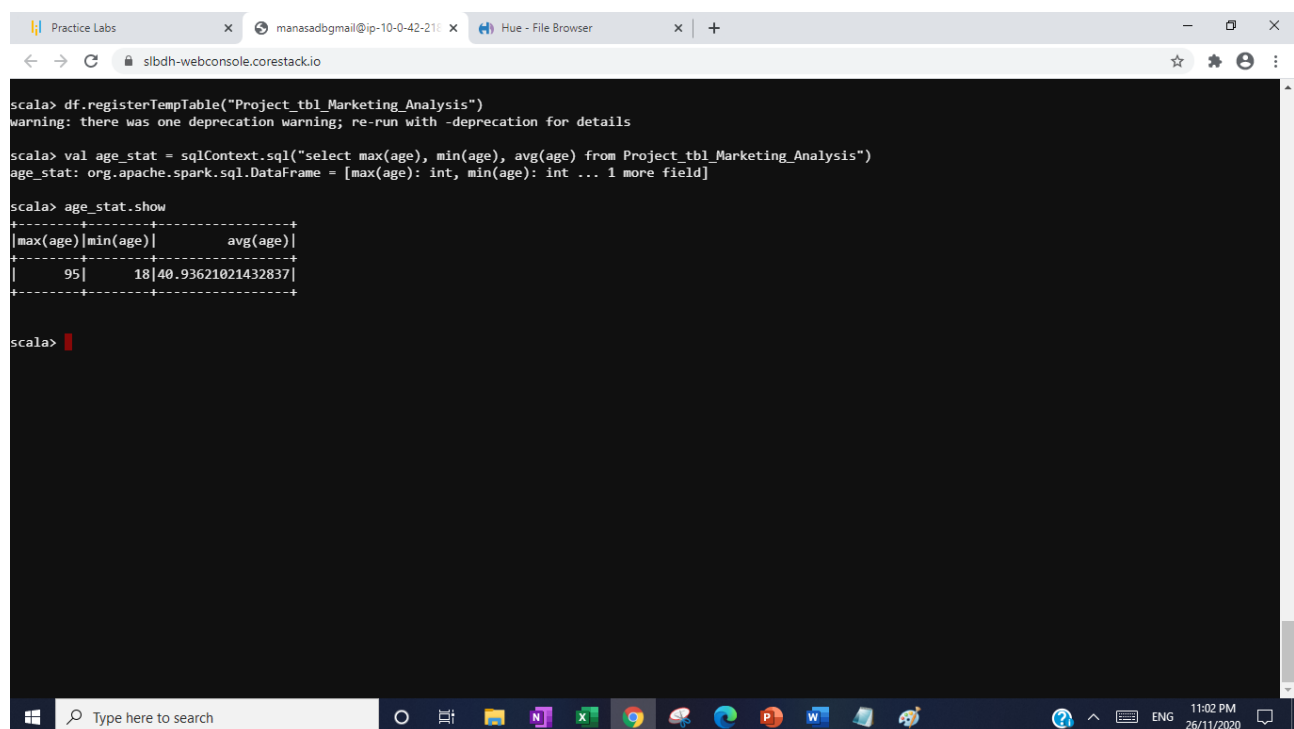
Task 3: Give the maximum, mean, and minimum age of the average targeted customer.

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

```
scala> val age_stat = sqlContext.sql("select max(age), min(age), avg(age) from Project_tbl_Marketing_Analysis")
```

```
scala> age_stat.show()
```

Output 3:



The screenshot shows a web browser window with a terminal interface. The terminal displays the following Scala code and its output:

```
scala> df.registerTempTable("Project_tbl_Marketing_Analysis")
warning: there was one deprecation warning; re-run with -deprecation for details

scala> val age_stat = sqlContext.sql("select max(age), min(age), avg(age) from Project_tbl_Marketing_Analysis")
age_stat: org.apache.spark.sql.DataFrame = [max(age): int, min(age): int ... 1 more field]

scala> age_stat.show
+-----+-----+-----+
|max(age)|min(age)|      avg(age)|
+-----+-----+-----+
|      95|      18|40.93621021432837|
+-----+-----+-----+

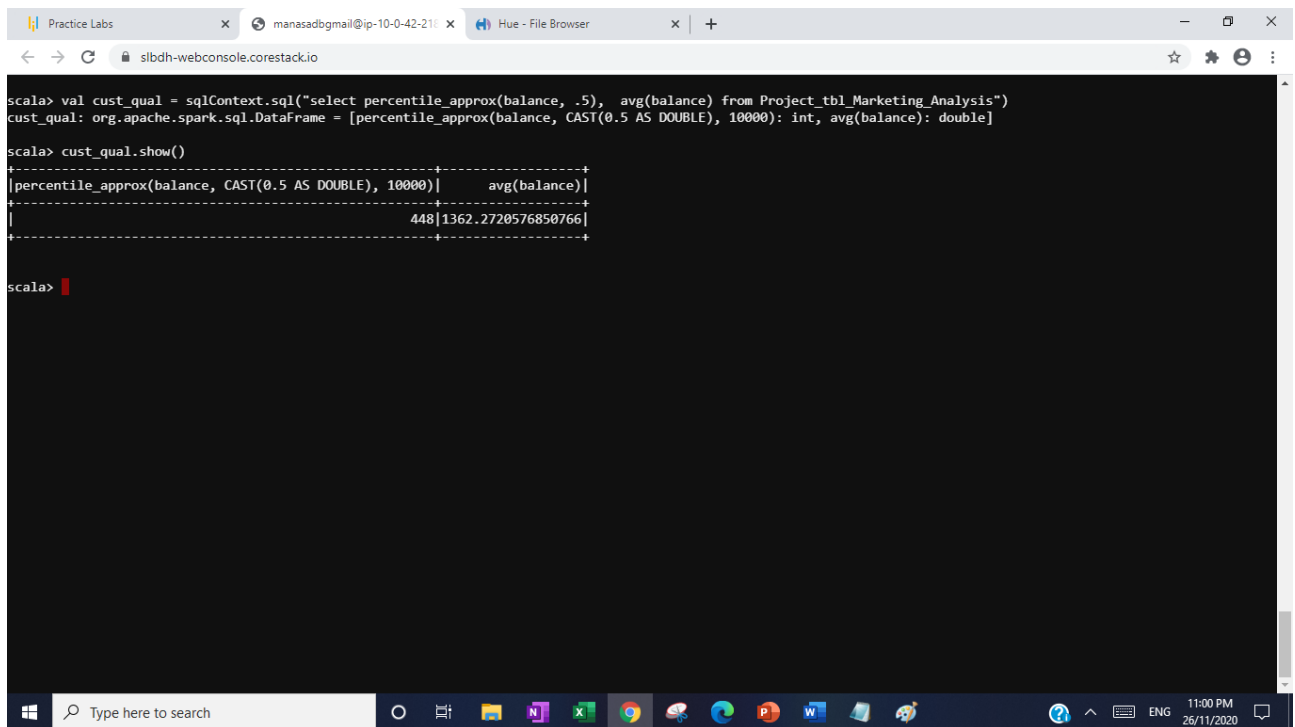
scala>
```

The output shows a table with three columns: max(age), min(age), and avg(age). The values are 95, 18, and 40.93621021432837 respectively.

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

```
scala> val cust_qual = sqlContext.sql("select percentile_approx(balance, .5), avg(balance)  
from Project_tbl_Marketing_Analysis")
```

```
scala> cust_qual.show()
```

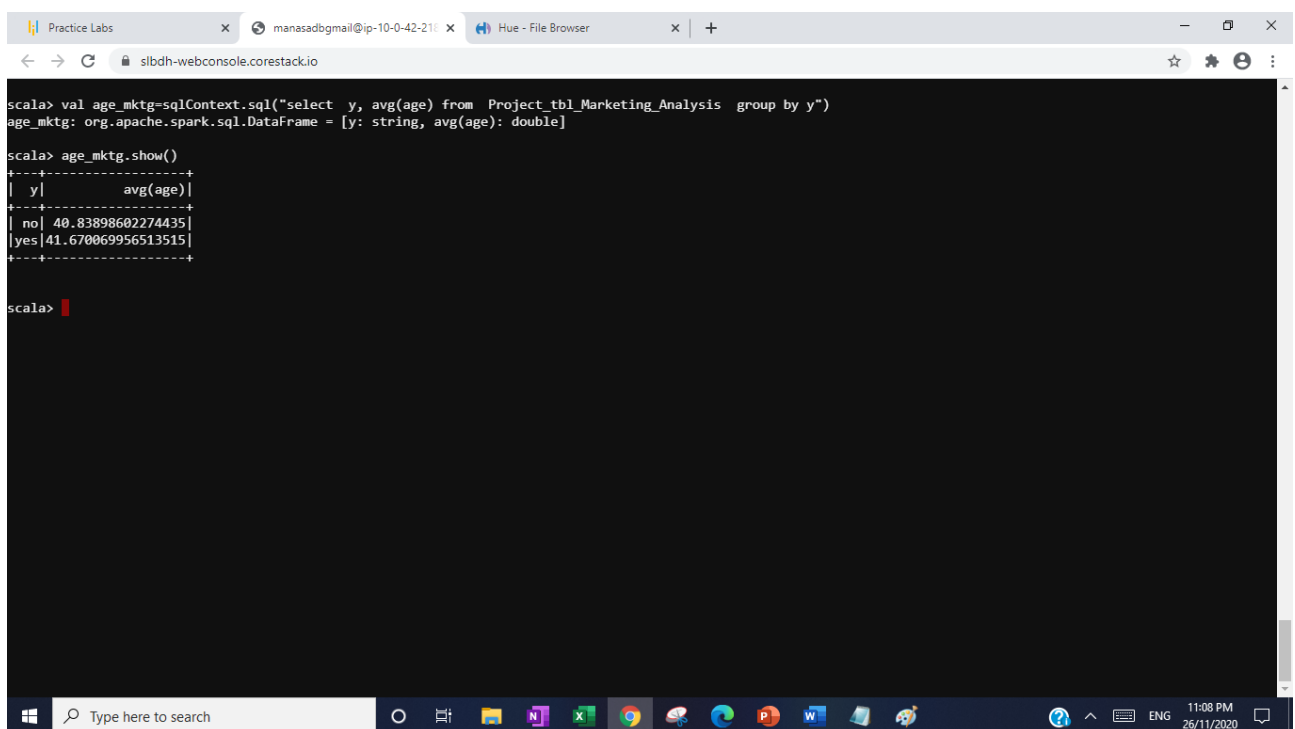
Output 4:

```
scala> val cust_qual = sqlContext.sql("select percentile_approx(balance, .5), avg(balance) from Project_tbl_Marketing_Analysis")  
cust_qual: org.apache.spark.sql.DataFrame = [percentile_approx(balance, CAST(0.5 AS DOUBLE), 10000): int, avg(balance): double]  
  
scala> cust_qual.show()  
+-----+-----+  
|percentile_approx(balance, CAST(0.5 AS DOUBLE), 10000)| avg(balance)|  
+-----+-----+  
|448|1362.2720576850766|  
+-----+-----+  
  
scala>
```

Task 5: Check if age matters in marketing subscription for deposit

```
scala> val age_mktg=sqlContext.sql("select y, avg(age) from Project_tbl_Marketing_Analysis group by y")
```

```
scala> age_mktg.show()
```

Output 5:

The screenshot shows a web browser window with a terminal interface. The terminal displays the following commands and output:

```
scala> val age_mktg=sqlContext.sql("select y, avg(age) from Project_tbl_Marketing_Analysis group by y")
age_mktg: org.apache.spark.sql.DataFrame = [y: string, avg(age): double]

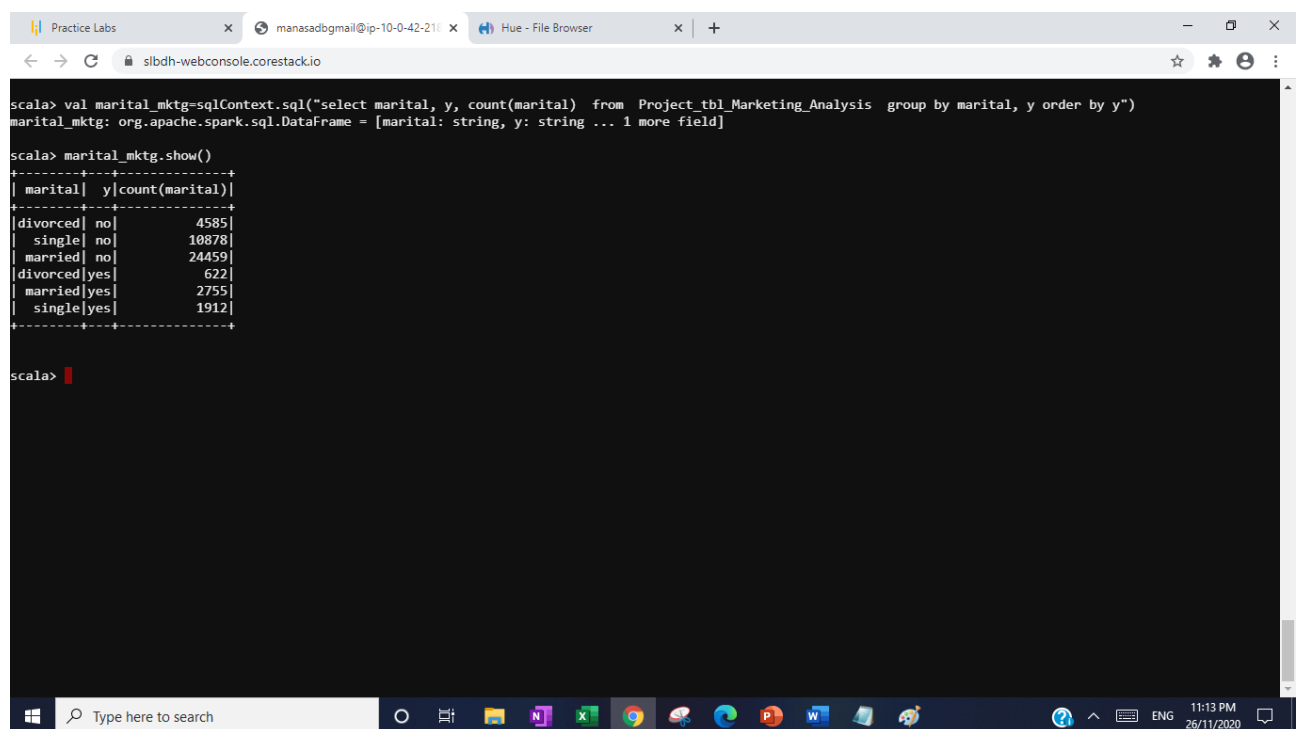
scala> age_mktg.show()
+-----+-----+
| y |      avg(age) |
+-----+-----+
| no | 40.83898602274435 |
| yes | 41.670069956513515 |
+-----+-----+
```

The output shows two rows of data. The first row is for 'no' with an average age of 40.83898602274435. The second row is for 'yes' with an average age of 41.670069956513515.

Task 6: Check if marital status mattered for a subscription to deposit

```
scala> val marital_mktg=sqlContext.sql("select marital, y, count(marital) from  
Project_tbl_Marketing_Analysis group by marital, y order by y")
```

```
scala> marital_mktg.show()
```

Output 6:

```
scala> val marital_mktg=sqlContext.sql("select marital, y, count(marital) from Project_tbl_Marketing_Analysis group by marital, y order by y")
marital_mktg: org.apache.spark.sql.DataFrame = [marital: string, y: string ... 1 more field]

scala> marital_mktg.show()
+-----+-----+-----+
| marital | y | count(marital) |
+-----+-----+-----+
| divorced | no | 4585 |
| single | no | 10878 |
| married | no | 24459 |
| divorced | yes | 622 |
| married | yes | 2755 |
| single | yes | 1912 |
+-----+-----+-----+

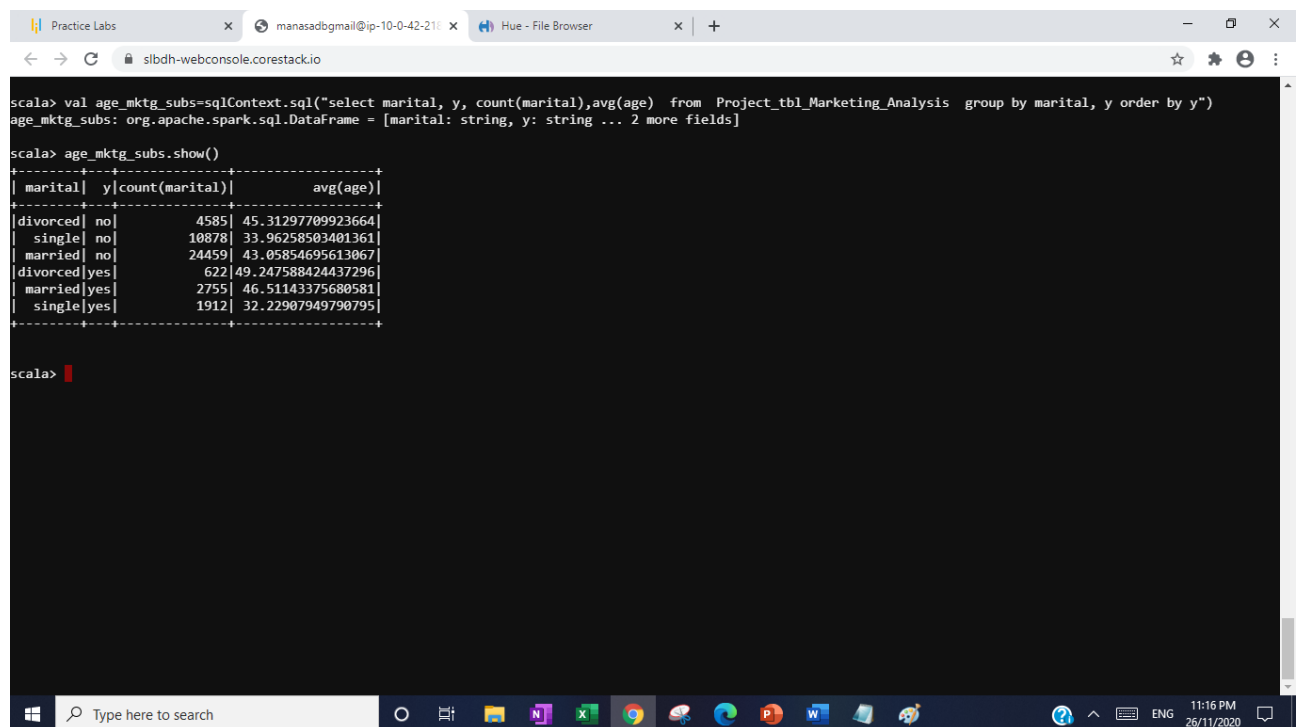
scala>
```

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

```
scala> val age_mktg_subs=sqlContext.sql("select marital, y, count(marital),avg(age) from Project_tbl_Marketing_Analysis group by marital, y order by y")
```

```
scala> age_mktg_subs.show()
```

Output 7:



The screenshot shows a web browser window with a terminal interface. The terminal displays the following Scala code and its output:

```
scala> val age_mktg_subs=sqlContext.sql("select marital, y, count(marital),avg(age) from Project_tbl_Marketing_Analysis group by marital, y order by y")
age_mktg_subs: org.apache.spark.sql.DataFrame = [marital: string, y: string ... 2 more fields]

scala> age_mktg_subs.show()
+-----+-----+-----+-----+
| marital | y | count(marital) | avg(age) |
+-----+-----+-----+-----+
| divorced | no | 4585 | 45.31297709923664 |
| single | no | 10878 | 33.96258503401361 |
| married | no | 24459 | 43.05854695613067 |
| divorced | yes | 622 | 49.247588424437296 |
| married | yes | 2755 | 46.51143375680581 |
| single | yes | 1912 | 32.22907949790795 |
+-----+-----+-----+-----+
```

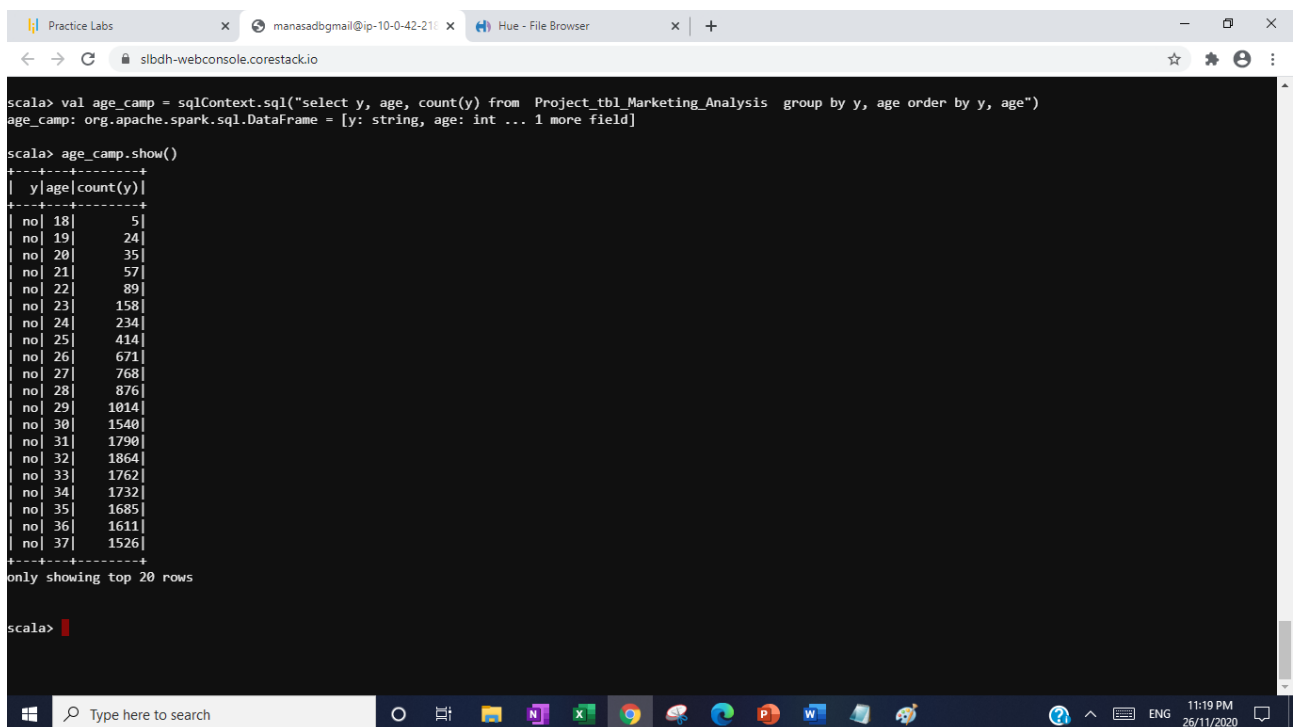
The output is a table with 4 columns: marital, y, count(marital), and avg(age). It shows data for six combinations of marital status and subscription type (y). The table is sorted by y.

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

```
scala> val age_camp = sqlContext.sql("select y, age, count(y) from  
Project_tbl_Marketing_Analysis group by y, age order by y, age")
```

```
scala> age_camp.show()
```

Output 8:



The screenshot shows a web browser window with a terminal interface. The terminal displays the following code and output:

```
scala> val age_camp = sqlContext.sql("select y, age, count(y) from Project_tbl_Marketing_Analysis group by y, age order by y, age")
age_camp: org.apache.spark.sql.DataFrame = [y: string, age: int ... 1 more field]

scala> age_camp.show()
+-----+
| y | age | count(y) |
+-----+
| no | 18 | 5 |
| no | 19 | 24 |
| no | 20 | 35 |
| no | 21 | 57 |
| no | 22 | 89 |
| no | 23 | 158 |
| no | 24 | 234 |
| no | 25 | 414 |
| no | 26 | 671 |
| no | 27 | 768 |
| no | 28 | 876 |
| no | 29 | 1014 |
| no | 30 | 1540 |
| no | 31 | 1790 |
| no | 32 | 1864 |
| no | 33 | 1762 |
| no | 34 | 1732 |
| no | 35 | 1685 |
| no | 36 | 1611 |
| no | 37 | 1526 |
+-----+
only showing top 20 rows

scala>
```

The output shows a table with three columns: 'y', 'age', and 'count(y)'. The data is sorted by 'y' and then 'age'. The 'y' column contains the value 'no' for all rows. The 'age' column ranges from 18 to 37. The 'count(y)' column shows the frequency of each age group. The output is truncated after 20 rows, with a message 'only showing top 20 rows'.