# Apache Spark—Real Time Project—Marketing Analysis from Bank Campaign Data

## Solution Code:

------------------------

The following commands need to be run on the Spark Shell of Scala by given order.

Note:

1. Input data file named "Bank_Data.csv" need to be placed in root HDFS directory.
2. Comment lines like "-- Solution: 1" should be skipped.
3. If required the result can be saved to HDFS directory using given save command (df2.write.format("com.databricks.spark.csv").save("Result"))

## Commands:

```
spark-shell --master local --packages com.databricks:spark-csv_2.10:1.3.0

case class Bank(age:String,job:String,marital:String,balance:String,y:String)

val rdd = sc.textFile("Bank_Data.csv")

val header = rdd.first()

val data = rdd.filter(row => row != header)

-- Solution: 1 (Creating Data Frame)

val df = data.map(x => x.replaceAll("\"","").split(";")).map(x=> Bank(x(0),x(1),x(2),x(5),x(16))).toDF()

df.show

df.registerTempTable("temp")

-- Solution: 2 (Subscription Success Rate)

val df2 = sqlContext.sql("select cast(avg(case when t.y == 'yes' then 1.0 else 0 end) as  decimal(2,2))
as Success_Rate from temp t")

df2.show
```

```
-- Solution: 2A (Subscription Failure Rate)

val df2 = sqlContext.sql("select cast(avg(case when t.y != 'yes' then 1.0 else 0 end) as  decimal(2,2)) as Failure_Rate from temp t")

df2.show

-- Solution: 3 (Maximum, Average, Minimum Age)

val df2 = sqlContext.sql("select max(age) as Maximum_Age, ( ( max(age) + min(age) ) / 2)  as Mean_Age , round(avg(age)) as Average_Age, min(age) as Minimum_Age from temp")

df2.show

-- Solution: 4 (Average and Median Balance)

val df2 = sqlContext.sql("SELECT balance, CAST(AVG(balance) OVER( ) as decimal(16,2)) AS Average_Balance, CAST(balance - AVG(balance) OVER ( ) as decimal(16,2) ) AS Difference FROM temp")

df2.show

df2.write.format("com.databricks.spark.csv").save("Result4")


-- Solution: 5 (Subscription by Age)

val df2 = sqlContext.sql("select age, count(*) as Subscription_Count from temp where  y = 'yes' group by age order by age")

df2.show

df2.write.format("com.databricks.spark.csv").save("Result5")


-- Solution: 6 (Subscription by Marital)

val df2 = sqlContext.sql("select marital, count(*) as Subscription_Count from temp where  y = 'yes' group by marital order by marital")

df2.show

-- Solution: 7 (Subscription by Age & Marital)

val df2 = sqlContext.sql("select age, marital, count(*) as Subscription_Count from temp where  y = 'yes' group by age,marital order by age,marital")

df2.show

df2.write.format("com.databricks.spark.csv").save("Result7")
```

-- Solution: 8 (Feature by Age)

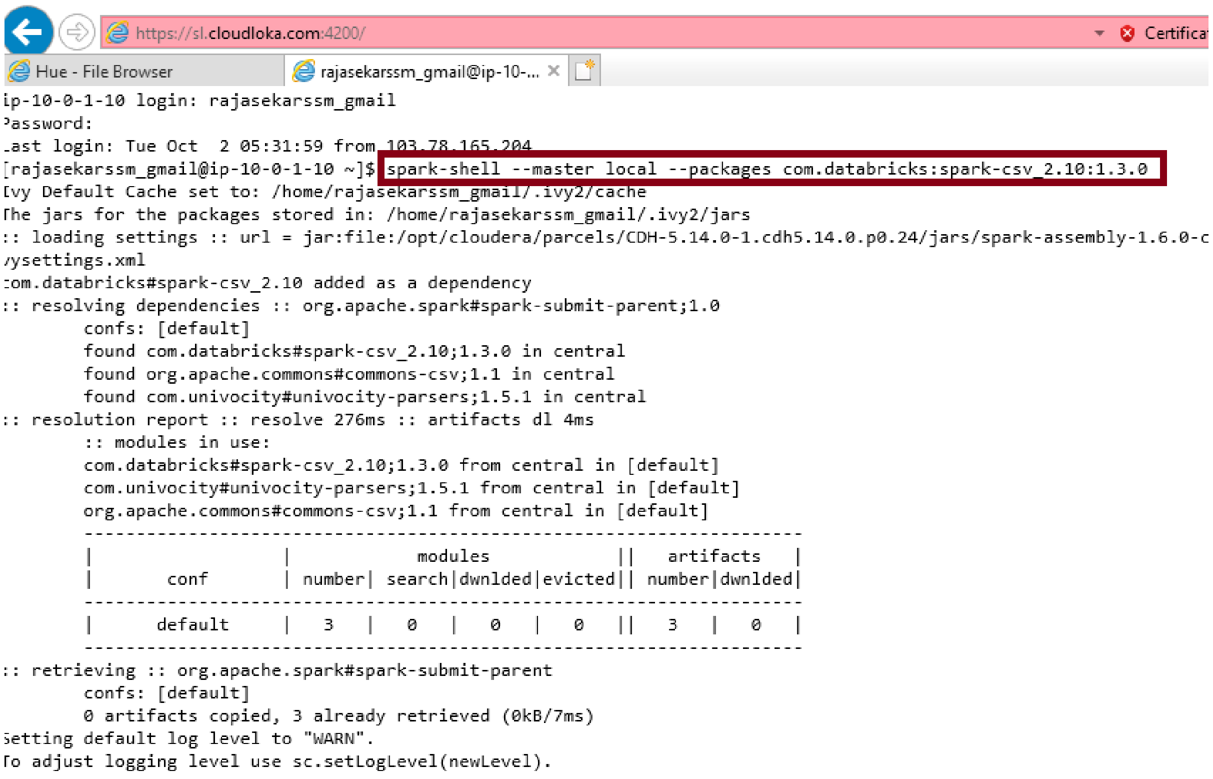
val df2 = sqlContext.sql("select sum(case when t.y == 'yes' and age >= 18 and age <= 30 then 1 else 0 end) as Subscriber_Age_18to30, sum(case when t.y == 'yes' and age >= 31 and age <= 60 then 1 else 0 end) as Subscriber_Age_31to60, sum(case when t.y == 'yes' and age > 60 then 1 else 0 end) as Subscriber_Age_Above60 from temp t")

df2.show

## Solution Screenshots:

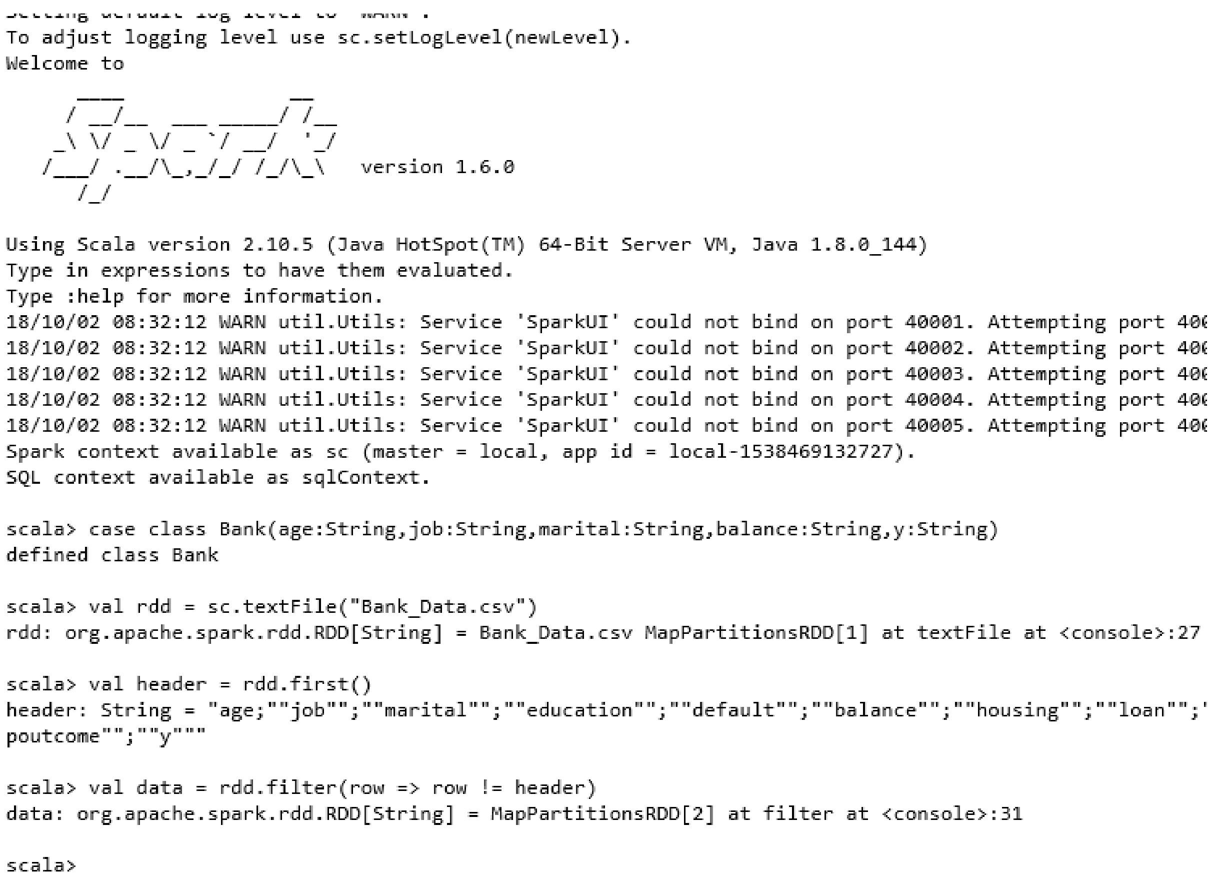
## -----------------------------

Screenshot 1:



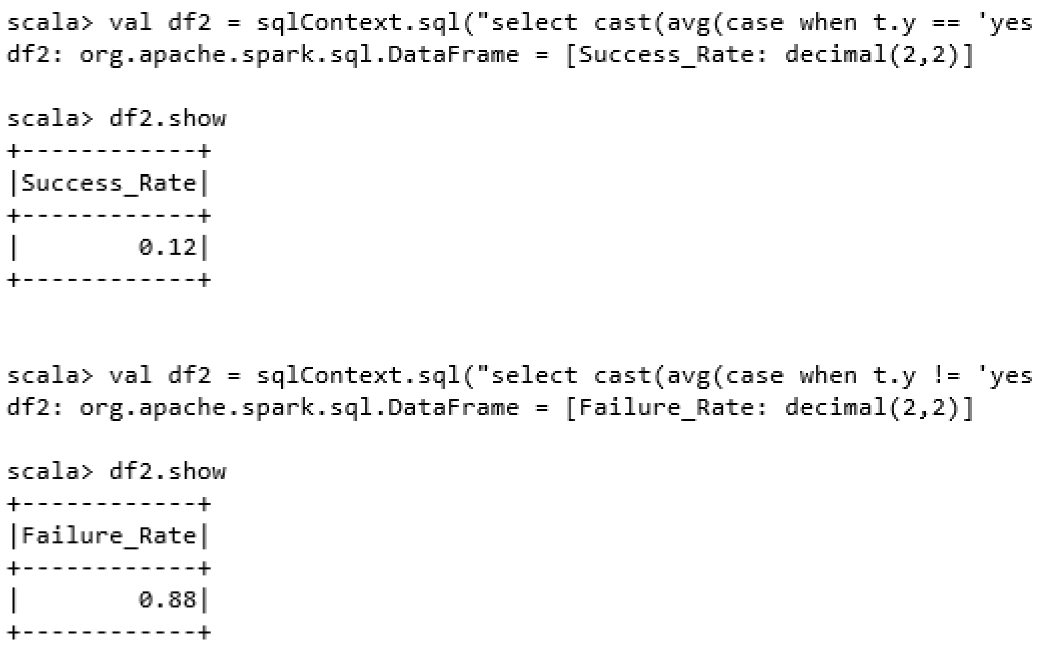
```
ip-10-0-1-10 login: rajasekarssm_gmail
Password:
.ast login: Tue Oct  2 05:31:59 from 103.78.165.204
[rajasekarssm_gmail@ip-10-0-1-10 ~]$ spark-shell --master local --packages com.databricks:spark-csv_2.10:1.3.0
Ivy Default Cache set to: /home/rajasekarssm_gmail/.ivy2/cache
The jars for the packages stored in: /home/rajasekarssm_gmail/.ivy2/jars
:: loading settings :: url = jar:file:/opt/cloudera/parcels/CDH-5.14.0-1.cdh5.14.0.p0.24/jars/spark-assembly-1.6.0-c
ivysettings.xml
com.databricks#spark-csv_2.10 added as a dependency
:: resolving dependencies :: org.apache.spark#spark-submit-parent;1.0
        confs: [default]
        found com.databricks#spark-csv_2.10;1.3.0 in central
        found org.apache.commons#commons-csv;1.1 in central
        found com.univocity#univocity-parsers;1.5.1 in central
:: resolution report :: resolve 276ms :: artifacts dl 4ms
        :: modules in use:
        com.databricks#spark-csv_2.10;1.3.0 from central in [default]
        com.univocity#univocity-parsers;1.5.1 from central in [default]
        org.apache.commons#commons-csv;1.1 from central in [default]
        ---------------------------------------------------------------------
        |                  |            modules            ||   artifacts   |
        |       conf       | number| search|dwnlded|evicted|| number|dwnlded|
        ---------------------------------------------------------------------
        |      default     |   3   |   0   |   0   |   0   ||   3   |   0   |
        ---------------------------------------------------------------------
:: retrieving :: org.apache.spark#spark-submit-parent
        confs: [default]
        0 artifacts copied, 3 already retrieved (0kB/7ms)
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel).
```

Screenshot 2:



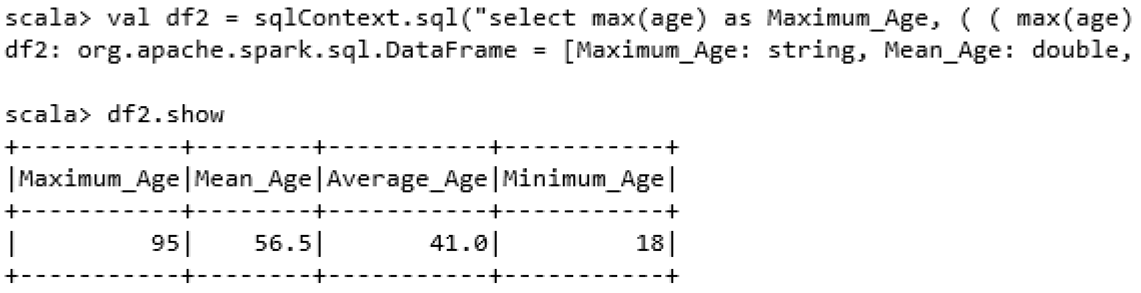
```
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel).
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 1.6.0
      /_/

Using Scala version 2.10.5 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_144)
Type in expressions to have them evaluated.
Type :help for more information.
18/10/02 08:32:12 WARN util.Utils: Service 'SparkUI' could not bind on port 40001. Attempting port 400
18/10/02 08:32:12 WARN util.Utils: Service 'SparkUI' could not bind on port 40002. Attempting port 400
18/10/02 08:32:12 WARN util.Utils: Service 'SparkUI' could not bind on port 40003. Attempting port 400
18/10/02 08:32:12 WARN util.Utils: Service 'SparkUI' could not bind on port 40004. Attempting port 400
18/10/02 08:32:12 WARN util.Utils: Service 'SparkUI' could not bind on port 40005. Attempting port 400
Spark context available as sc (master = local, app id = local-1538469132727).
SQL context available as sqlContext.

scala> case class Bank(age:String,job:String,marital:String,balance:String,y:String)
defined class Bank

scala> val rdd = sc.textFile("Bank_Data.csv")
rdd: org.apache.spark.rdd.RDD[String] = Bank_Data.csv MapPartitionsRDD[1] at textFile at <console>:27

scala> val header = rdd.first()
header: String = "age;""job"";""marital"";""education"";""default"";""balance"";""housing"";""loan"";'
poutcome"";""y"""

scala> val data = rdd.filter(row => row != header)
data: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[2] at filter at <console>:31

scala>
```

Screenshot 3:



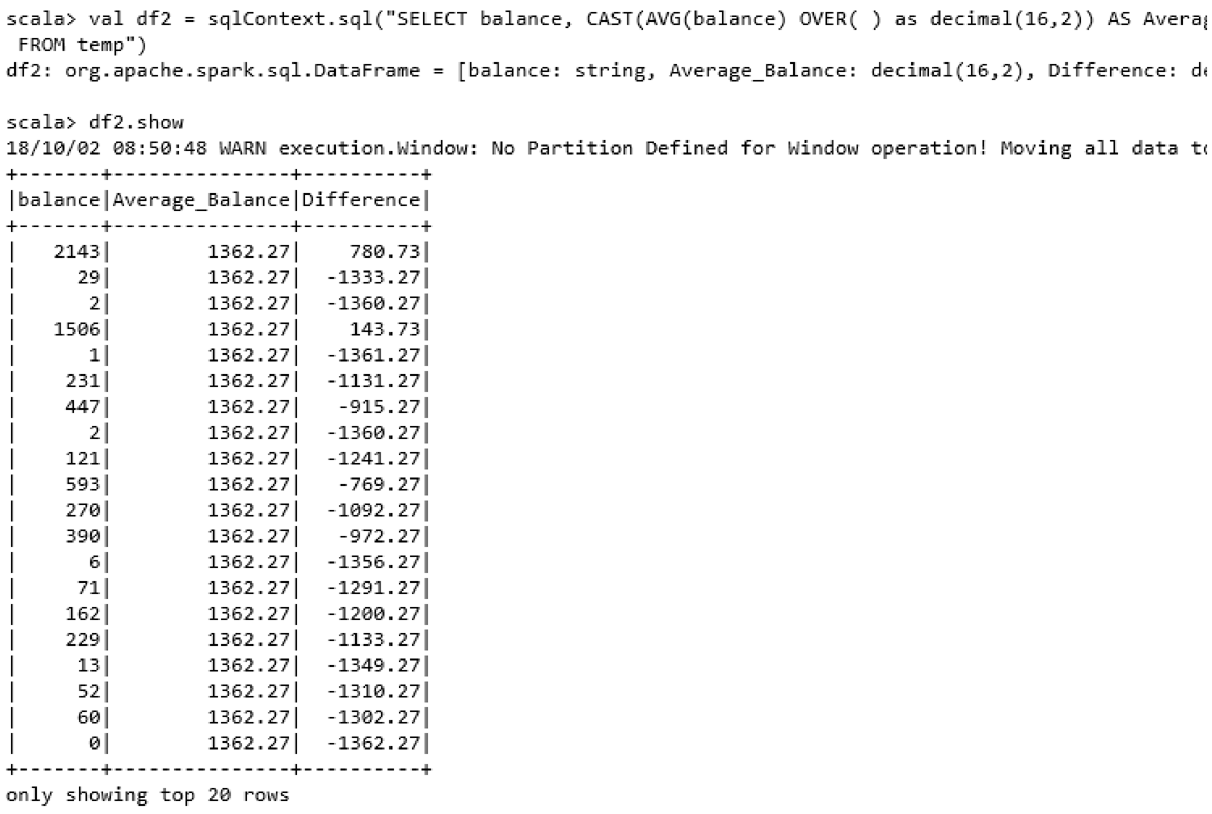
```
scala> val df2 = sqlContext.sql("select cast(avg(case when t.y == 'yes
df2: org.apache.spark.sql.DataFrame = [Success_Rate: decimal(2,2)]

scala> df2.show
+------------+
|Success_Rate|
+------------+
|        0.12|
+------------+


scala> val df2 = sqlContext.sql("select cast(avg(case when t.y != 'yes
df2: org.apache.spark.sql.DataFrame = [Failure_Rate: decimal(2,2)]

scala> df2.show
+------------+
|Failure_Rate|
+------------+
|        0.88|
+------------+
```

Screenshot 4:

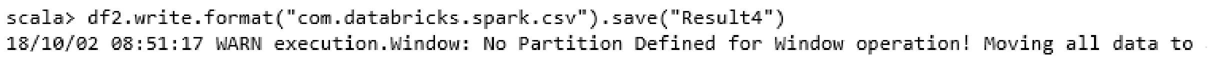


```
scala> val df2 = sqlContext.sql("select max(age) as Maximum_Age, ( ( max(age)
df2: org.apache.spark.sql.DataFrame = [Maximum_Age: string, Mean_Age: double,

scala> df2.show
+-----------+--------+-----------+-----------+
|Maximum_Age|Mean_Age|Average_Age|Minimum_Age|
+-----------+--------+-----------+-----------+
|         95|    56.5|       41.0|         18|
+-----------+--------+-----------+-----------+
```

Screenshot 5:

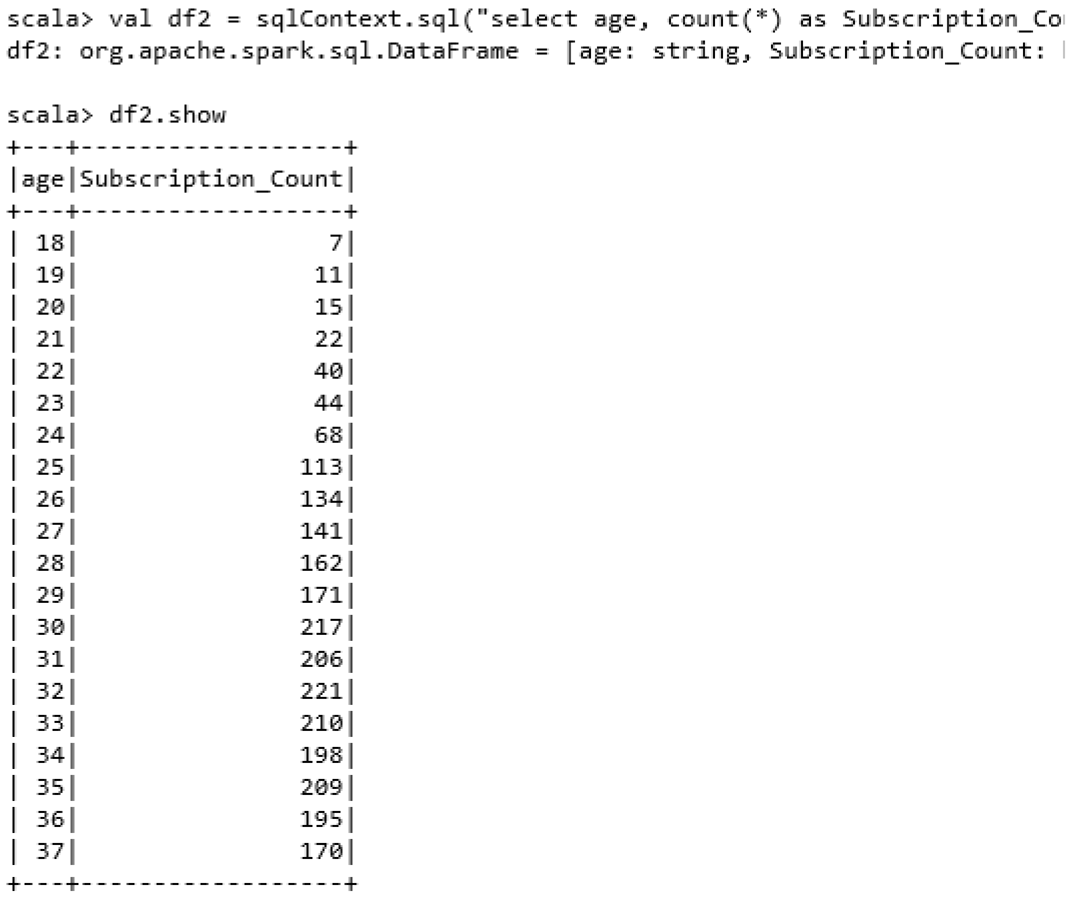


```
scala> val df2 = sqlContext.sql("SELECT balance, CAST(AVG(balance) OVER( ) as decimal(16,2)) AS Avera
 FROM temp")
df2: org.apache.spark.sql.DataFrame = [balance: string, Average_Balance: decimal(16,2), Difference: d

scala> df2.show
18/10/02 08:50:48 WARN execution.Window: No Partition Defined for Window operation! Moving all data t
+-------+---------------+----------+
|balance|Average_Balance|Difference|
+-------+---------------+----------+
|   2143|        1362.27|    780.73|
|     29|        1362.27|  -1333.27|
|      2|        1362.27|  -1360.27|
|   1506|        1362.27|    143.73|
|      1|        1362.27|  -1361.27|
|    231|        1362.27|  -1131.27|
|    447|        1362.27|   -915.27|
|      2|        1362.27|  -1360.27|
|    121|        1362.27|  -1241.27|
|    593|        1362.27|   -769.27|
|    270|        1362.27|  -1092.27|
|    390|        1362.27|   -972.27|
|      6|        1362.27|  -1356.27|
|     71|        1362.27|  -1291.27|
|    162|        1362.27|  -1200.27|
|    229|        1362.27|  -1133.27|
|     13|        1362.27|  -1349.27|
|     52|        1362.27|  -1310.27|
|     60|        1362.27|  -1302.27|
|      0|        1362.27|  -1362.27|
+-------+---------------+----------+
only showing top 20 rows
```



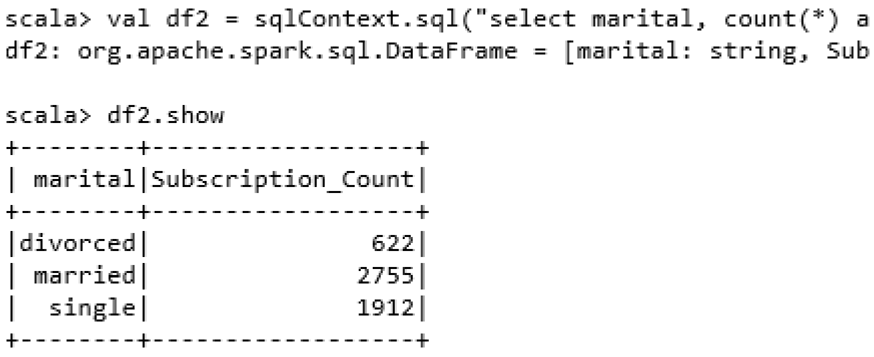
```
scala> df2.write.format("com.databricks.spark.csv").save("Result4")
18/10/02 08:51:17 WARN execution.Window: No Partition Defined for Window operation! Moving all data to
```

Screenshot 6:



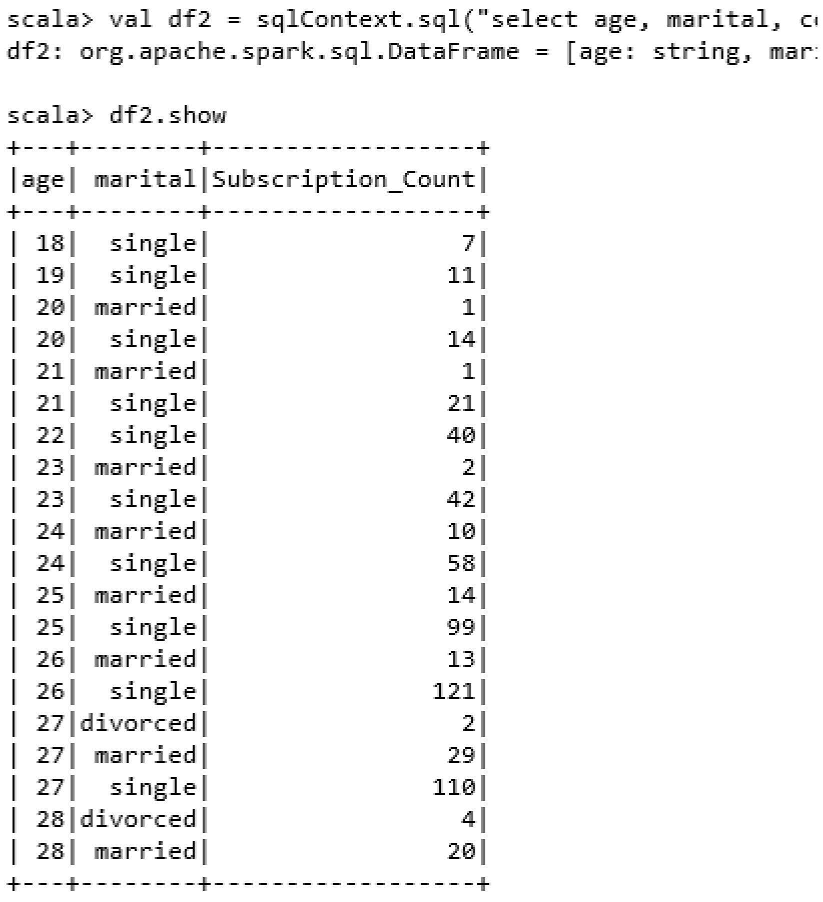
```
scala> val df2 = sqlContext.sql("select age, count(*) as Subscription_Co
df2: org.apache.spark.sql.DataFrame = [age: string, Subscription_Count:

scala> df2.show
+---+------------------+
|age|Subscription_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|
+---+------------------+
```

Screenshot 7:



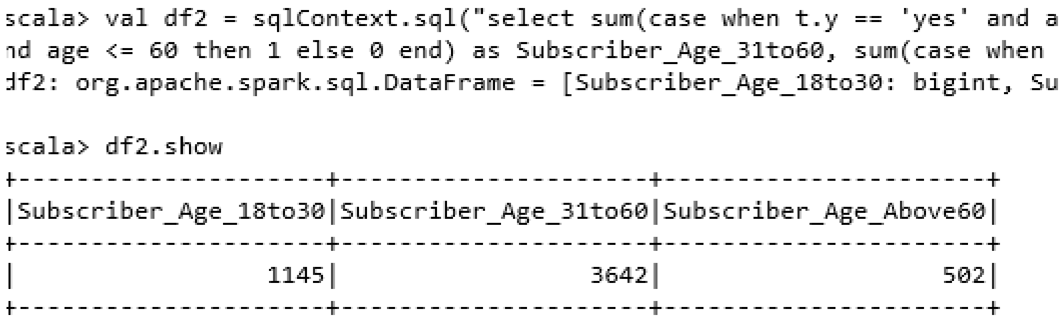
```
scala> val df2 = sqlContext.sql("select marital, count(*) a
df2: org.apache.spark.sql.DataFrame = [marital: string, Sub

scala> df2.show
+--------+------------------+
| marital|Subscription_Count|
+--------+------------------+
|divorced|               622|
| married|              2755|
|  single|              1912|
+--------+------------------+
```

Screenshot 8:



```
scala> val df2 = sqlContext.sql("select age, marital, c
df2: org.apache.spark.sql.DataFrame = [age: string, mar

scala> df2.show
+---+--------+------------------+
|age| marital|Subscription_Count|
+---+--------+------------------+
| 18|  single|                 7|
| 19|  single|                11|
| 20| married|                 1|
| 20|  single|                14|
| 21| married|                 1|
| 21|  single|                21|
| 22|  single|                40|
| 23| married|                 2|
| 23|  single|                42|
| 24| married|                10|
| 24|  single|                58|
| 25| married|                14|
| 25|  single|                99|
| 26| married|                13|
| 26|  single|               121|
| 27|divorced|                 2|
| 27| married|                29|
| 27|  single|               110|
| 28|divorced|                 4|
| 28| married|                20|
+---+--------+------------------+
```

Screenshot 9:



```
scala> val df2 = sqlContext.sql("select sum(case when t.y == 'yes' and a
nd age <= 60 then 1 else 0 end) as Subscriber_Age_31to60, sum(case when
df2: org.apache.spark.sql.DataFrame = [Subscriber_Age_18to30: bigint, Su

scala> df2.show
+---------------------+---------------------+----------------------+
|Subscriber_Age_18to30|Subscriber_Age_31to60|Subscriber_Age_Above60|
+---------------------+---------------------+----------------------+
|                 1145|                 3642|                   502|
+---------------------+---------------------+----------------------+
```