//1. load data and create spark data frame

val df=spark.read.format("csv").option("header","true")
.option("inferschema","true").option("delimiter",";")
.load("C:\\DataScienceClass\\Apache Spark\\Projects\\myproject\\bank-full.csv")

df.printSchema();

```
Type :help for more information.

scala> val df=spark.read.format("csv").option("header","true").option("inferschema","true"
df: org.apache.spark.sql.DataFrame = [age: int, job: string ... 15 more fields]

scala> 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)


scala>
```

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

val success_rate = df.filter($"y" === "yes").count().toDouble/df.count().toDouble

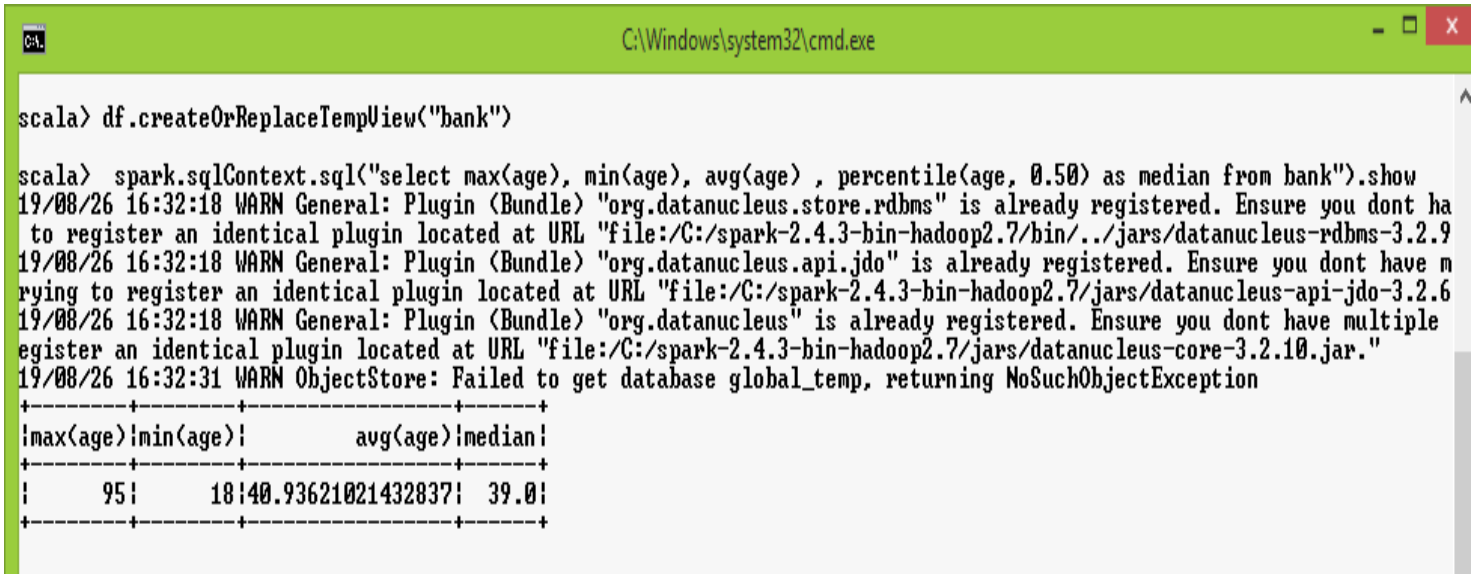
```
scala>   val success_rate = df.filter($"y" === "yes").count().toDouble/df.count().toDouble
success_rate: Double = 0.11698480458295547
```

marketing success rate = 0.117

//3. Check max, min, Mean and median age of average targeted customer

df.createOrReplaceTempView("bank")

spark.sqlContext.sql("select max(age), min(age), avg(age) , percentile(age, 0.50) from bank").show



Mean age=41

Max age =95

Min age =18

Median age=39

//4. Check quality of clients by checking average balance, median balance of clients

spark.sql("select mean(balance), percentile_approx(balance,0.5) from bank").show

```
scala> spark.sql("select mean(balance), percentile_approx(balance,0.5) as median from bank").show
+------------------+------+
|      avg(balance)|median|
+------------------+------+
|1362.2720576850766|   448|
+------------------+------+
```

Average balance =1362.27

Median Balance =448

//5. Check if age matters in marketing subscription for deposit

df.groupBy("age","y").count.orderBy('age).show()

df.select(countDistinct($"age")).show //77

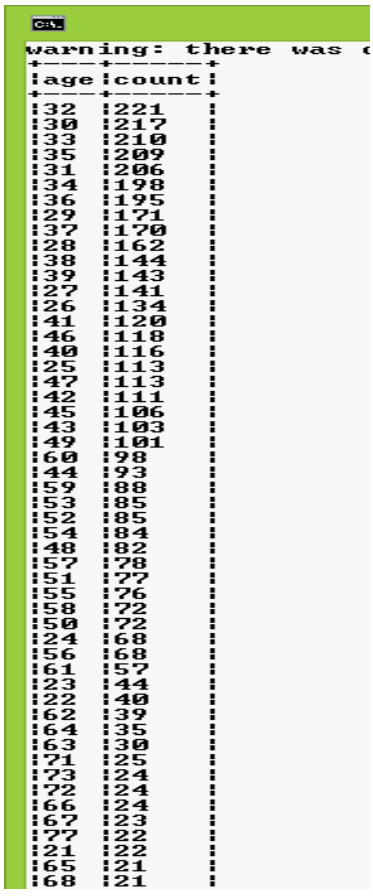
df.filter($"y"==="yes").groupBy("age").count.orderBy('count desc).show(77,false)

C:\Windows\system32\cmd.exe

```
scala> df.groupBy("age","y").count.orderBy('age).show()
+---+---+-----+
|age|  y|count|
+---+---+-----+
| 18| no|    5|
| 18|yes|    7|
| 19| no|   24|
| 19|yes|   11|
| 20| no|   35|
| 20|yes|   15|
| 21| no|   57|
| 21|yes|   22|
| 22|yes|   40|
| 22| no|   89|
| 23| no|  158|
| 23|yes|   44|
| 24|yes|   68|
| 24| no|  234|
| 25| no|  414|
| 25|yes|  113|
| 26|yes|  134|
| 26| no|  671|
| 27|yes|  141|
| 27| no|  768|
+---+---+-----+
only showing top 20 rows

scala>   df.select(countDistinct($"age")).show     //77
+----------------------+
|count(DISTINCT age)|
+----------------------+
|                    77|
+----------------------+

scala>   df.filter($"y"==="yes").groupBy("age").count.orderBy('count desc).show(77,false)
warning: there was one feature warning; re-run with -feature for details
+---+-----+
|age|count|
+---+-----+
|32 |221  |
|30 |217  |
|33 |210  |
|35 |209  |
|31 |206  |
|34 |198  |
|36 |195  |
|29 |171  |
|37 |170  |
|28 |162  |
|38 |144  |
|39 |143  |
|27 |141  |
```



Yes , age matters for the subscription. Mostly people of age around 25-37 are subscribed.

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

df.groupBy("marital","y").count.orderBy('marital).show()

df.filter($"y"==="yes").groupBy("marital").count.orderBy('count desc).show()

```
scala> :paste
// Entering paste mode (ctrl-D to finish)

df.groupBy("marital","y").count.orderBy('marital).show()
  df.filter($"y"==="yes").groupBy("marital").count.orderBy('count desc).show()

// Exiting paste mode, now interpreting.

warning: there was one feature warning; re-run with -feature for details
+--------+---+-----+
| marital|  y|count|
+--------+---+-----+
|divorced| no| 4585|
|divorced|yes|  622|
| married|yes| 2755|
| married| no|24459|
|  single| no|10878|
|  single|yes| 1912|
+--------+---+-----+

+--------+-----+
| marital|count|
+--------+-----+
| married| 2755|
|  single| 1912|
|divorced|  622|
+--------+-----+
```

Yes, marital status also matters .Mostly married couples are subscribed.

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

df.filter($"y"==="yes").groupBy("marital","age").count.orderBy('count desc).show(false)

```
scala> df.filter($"y"==="yes").groupBy("marital","age").count.orderBy('count desc).show(40,false)
warning: there was one feature warning; re-run with -feature for details
+-------+---+-----+
|marital|age|count|
+-------+---+-----+
|single |30 |151  |
|single |28 |138  |
|single |29 |133  |
|single |32 |124  |
|single |26 |121  |
|married|34 |118  |
|single |31 |111  |
|single |27 |110  |
|married|35 |101  |
|married|36 |100  |
|single |25 |99   |
|married|37 |98   |
|married|33 |97   |
|single |33 |97   |
|married|32 |87   |
|married|39 |87   |
|married|38 |86   |
|single |35 |84   |
|married|47 |83   |
|married|46 |80   |
|married|31 |80   |
|married|60 |73   |
|married|40 |73   |
|married|41 |72   |
|single |36 |71   |
|married|49 |71   |
|married|42 |70   |
|single |34 |69   |
|married|45 |68   |
|married|52 |67   |
|married|59 |66   |
|married|43 |62   |
|married|53 |60   |
|married|51 |59   |
|married|30 |59   |
|married|57 |58   |
|single |24 |58   |
|single |37 |57   |
|married|50 |57   |
|married|58 |54   |
+-------+---+-----+
only showing top 40 rows
```

Single people around the age 25-37 shows most subscriptions

//8. Do Feature engineering for age column and find right age effect on campaign

df.withColumn("age_category", when($"age" < 25 , "young").otherwise(when($"age" > 55 , "old").otherwise("mid_age"))).groupBy("age_category","y").count.show()

```
scala>  df.withColumn("age_category", when($"age" < 25 , "young").otherwise( when($"age" > 55 , "old").otherwise("mid_
+------------+---+-----+
|age_category|  y|count|
+------------+---+-----+
|     mid_age| no|35326|
|     mid_age|yes| 4176|
|       young| no|  602|
|         old|yes|  906|
|         old| no| 3994|
|       young|yes|  207|
+------------+---+-----+
```

We can conclude from the Feature Engineering that the 'Middle Aged' people age between 25 and 55 who should be targeted customers as they subscribe the most.