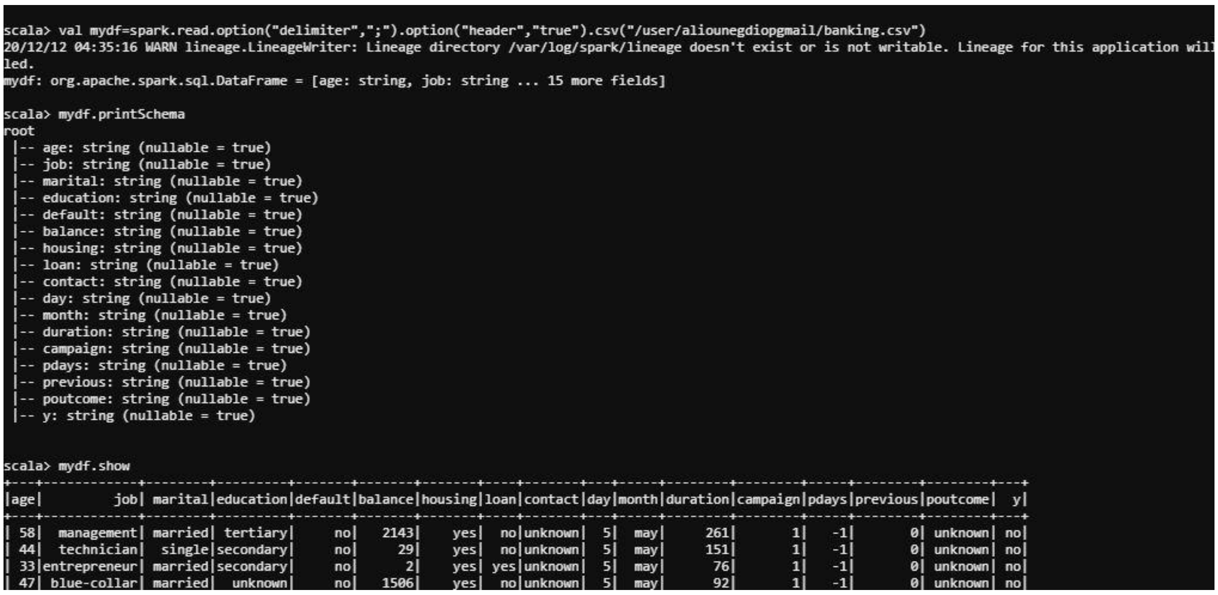
Market Analysis on SCALA

Question1: Load data and create a Spark data frame

val mydf=spark.read.option("delimiter",";").option("header","true").csv("/user/aliounegdiopgmail/banking.csv")



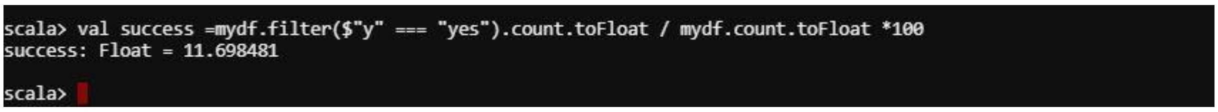
```
scala> val mydf=spark.read.option("delimiter",";").option("header","true").csv("/user/aliounegdiopgmail/banking.csv")
20/12/12 04:35:16 WARN lineage.LineageWriter: Lineage directory /var/log/spark/lineage doesn't exist or is not writable. Lineage for this application will
led.
mydf: org.apache.spark.sql.DataFrame = [age: string, job: string ... 15 more fields]

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


scala> mydf.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|
```

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

Success rate:

scala> val success =mydf.filter($"y" === "yes").count.toFloat / mydf.count.toFloat *100

success: Float = 11.698481

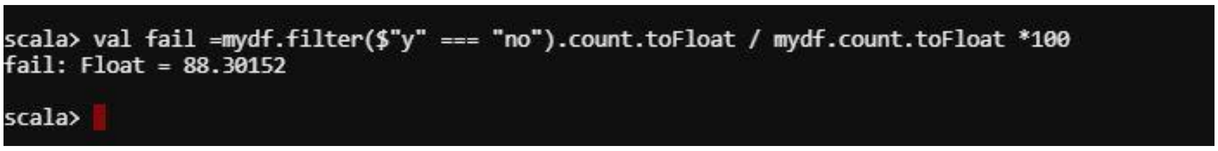


```
scala> val success =mydf.filter($"y" === "yes").count.toFloat / mydf.count.toFloat *100
success: Float = 11.698481

scala> 
```

Failure Rate:

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

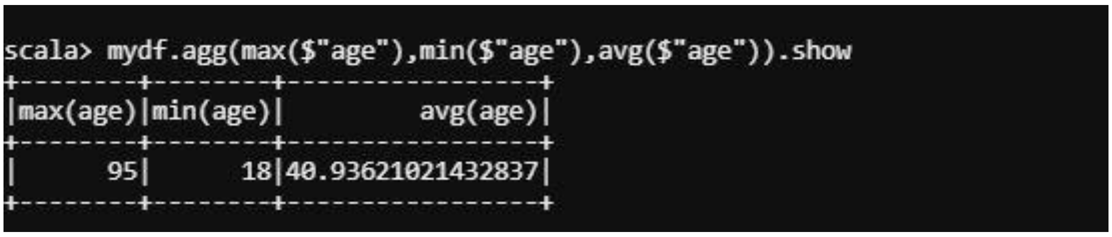
fail: Float = 88.30152



```
scala> val fail =mydf.filter($"y" === "no").count.toFloat / mydf.count.toFloat *100
fail: Float = 88.30152

scala> 
```

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

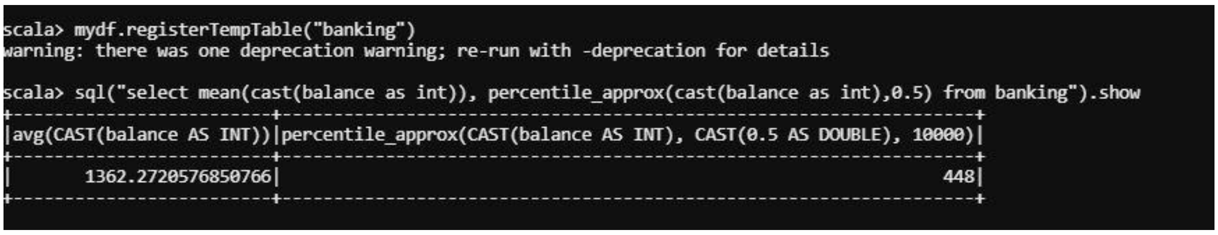
```
scala> mydf.agg(max($"age"),min($"age"),avg($"age")).show

+--------+--------+-----------------+

|max(age)|min(age)|         avg(age)|

+--------+--------+-----------------+

|      95|      18|40.93621021432837|

+--------+--------+-----------------+
```



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

Question4: Check the quality of customers by checking average balance, median balance of customers

```
scala> mydf.registerTempTable("banking")

warning: there was one deprecation warning; re-run with -deprecation for details

scala> sql("select mean(cast(balance as int)), percentile_approx(cast(balance as int),0.5) from banking").show

+------------------------+-----------------------------------------------------------------+

|avg(CAST(balance AS INT))|percentile_approx(CAST(balance AS INT), CAST(0.5 AS DOUBLE), 10000)|

+------------------------+-----------------------------------------------------------------+

|      1362.2720576850766|                                                              448|

+------------------------+-----------------------------------------------------------------+
```



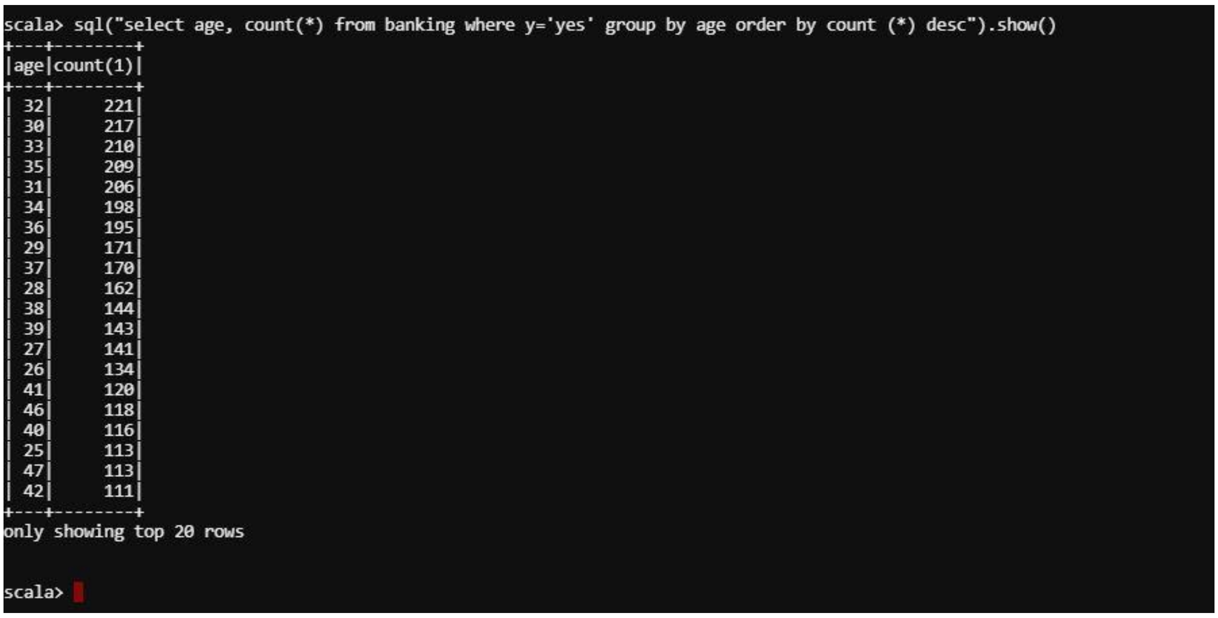
```
scala> mydf.registerTempTable("banking")
warning: there was one deprecation warning; re-run with -deprecation for details

scala> sql("select mean(cast(balance as int)), percentile_approx(cast(balance as int),0.5) from banking").show
+-------------------------+-------------------------------------------------------------------+
|avg(CAST(balance AS INT))|percentile_approx(CAST(balance AS INT), CAST(0.5 AS DOUBLE), 10000)|
+-------------------------+-------------------------------------------------------------------+
|       1362.2720576850766|                                                                448|
+-------------------------+-------------------------------------------------------------------+
```

Question5: Check if age matters in marketing subscription for deposit

```
scala> sql("select age, count(*) from banking where y='yes' group by age order by count (*) desc").show()
```

```
+---+--------+
|age|count(1)|
+---+--------+
| 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|
| 26|     134|
| 41|     120|
| 46|     118|
| 40|     116|
| 25|     113|
| 47|     113|
| 42|     111|
+---+--------+
```

only showing top 20 rows

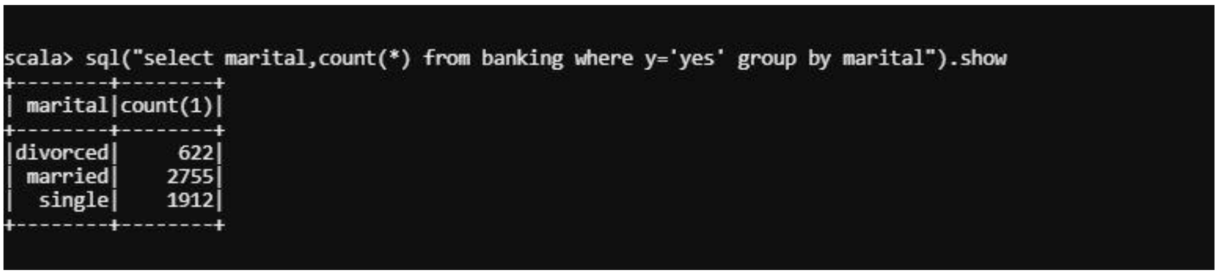


```
scala> sql("select age, count(*) from banking where y='yes' group by age order by count (*) desc").show()
+---+--------+
|age|count(1)|
+---+--------+
| 32|     221|
| 30|     217|
| 33|     210|
| 35|     209|
| 31|     206|
| 34|     198|
| 36|     195|
| 29|     171|
| 37|     170|
| 28|     162|
| 38|     144|
| 39|     143|
| 27|     141|
| 26|     134|
| 41|     120|
| 46|     118|
| 40|     116|
| 25|     113|
| 47|     113|
| 42|     111|
+---+--------+
only showing top 20 rows

scala>
```

Question6: Check if marital status mattered for a subscription to deposit

scala> sql("select marital,count(*) from banking where y='yes' group by marital").show

```
+--------+--------+
| marital|count(1)|
+--------+--------+
|divorced|     622|
| married|    2755|
|  single|    1912|
+--------+--------+
```



```
scala> sql("select marital,count(*) from banking where y='yes' group by marital").show
+--------+--------+
| marital|count(1)|
+--------+--------+
|divorced|     622|
| married|    2755|
|  single|    1912|
+--------+--------+
```

Question7: Check if age and marital status together mattered for a subscription to deposit scheme

scala> sql("select age, marital, count(*) as number from banking where y='yes' group by age,marital order by number desc ").show()

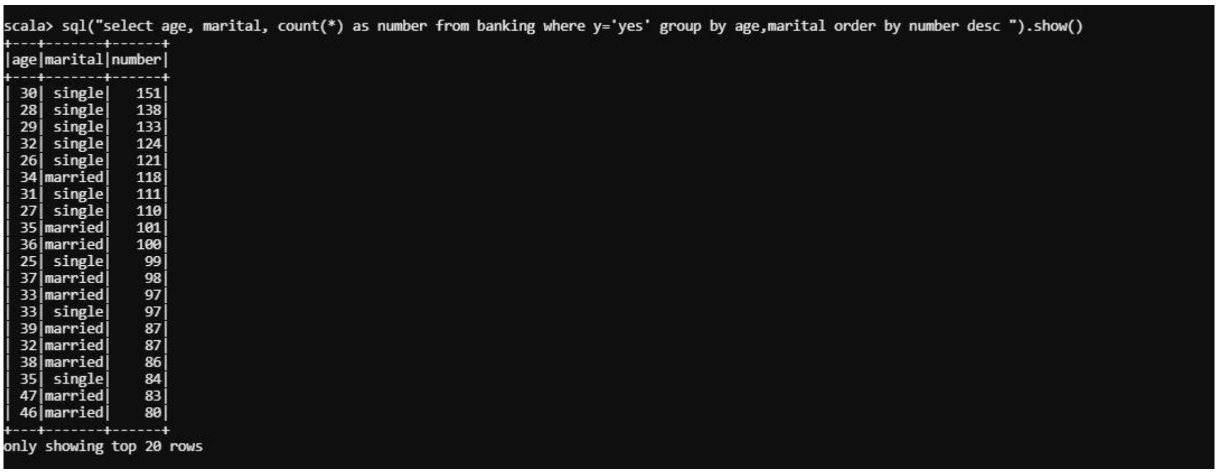
+---+-------+------+ |age|marital|number|+---+-------+------+

| 30| single| 151|

| 28| single| 138|

| 29| single| 133|

| 32| single| 124|

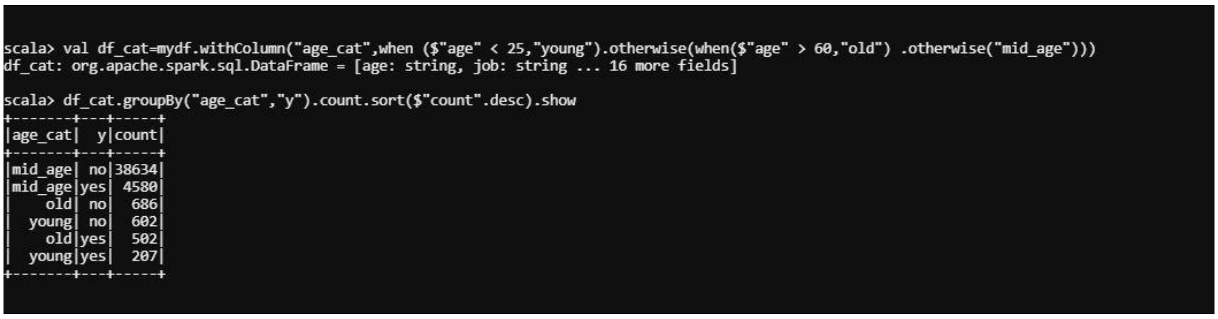


```
scala> sql("select age, marital, count(*) as number from banking where y='yes' group by age,marital order by number desc ").show()
+---+-------+------+
|age|marital|number|
+---+-------+------+
| 30| single|   151|
| 28| single|   138|
| 29| single|   133|
| 32| single|   124|
| 26| single|   121|
| 34|married|   118|
| 31| single|   111|
| 27| single|   110|
| 35|married|   101|
| 36|married|   100|
| 25| single|    99|
| 37|married|    98|
| 33|married|    97|
| 33| single|    97|
| 39|married|    87|
| 32|married|    87|
| 38|married|    86|
| 35| single|    84|
| 47|married|    83|
| 46|married|    80|
+---+-------+------+
only showing top 20 rows
```

Question8: Do feature engineering for the bank and find the right age effect on the campaign.

sql("select age,marital,count(*) from banking where y='yes' group by marital age").show

sql("select age, marital, count(*) as number from bank where y='yes' group by age,marital order by number desc ").show()



```
scala> val df_cat=mydf.withColumn("age_cat",when ($"age" < 25,"young").otherwise(when($"age" > 60,"old") .otherwise("mid_age")))
df_cat: org.apache.spark.sql.DataFrame = [age: string, job: string ... 16 more fields]

scala> df_cat.groupBy("age_cat","y").count.sort($"count".desc).show
+-------+---+-----+
|age_cat|  y|count|
+-------+---+-----+
|mid_age| no|38634|
|mid_age|yes| 4580|
|    old| no|  686|
|  young| no|  602|
|    old|yes|  502|
|  young|yes|  207|
+-------+---+-----+
```