Market Analysis in Banking Domain

by Sujit Sonar

Objective: Marketing analysis of the data generated by this campaign.

Q1) Load data and create a Spark data frame

rdd1=spark.sparkContext.textFile("/user/sujitsonargmail/spark_project/market_analysis_data.csv")
rdd2 = rdd1.map(lambda x: x.replace("\"",""))
header=rdd2.first()
header_list= header.split(";")
rdd3 = rdd2.filter(lambda x:x!= header) ## remove the header row
rdd4=rdd3.map(lambda x: x.split(";"))
df=rdd4.toDF(schema=header_list)
df.show()

```
>>> rdd1=spark.sparkContext.textFile("/user/sujitsonargmail/spark_project/market_analysis_data.csv")
>>> rdd2 = rdd1.map(lambda x: x.replace("\"",""))
>>> header=rdd2.first()
>>> header_list= header.split(";")
>>> rdd3 = rdd2.filter(lambda x:x!= header) ## remove the header row
>>> rdd4=rdd3.map(lambda x: x.split(";"))
>>> df=rdd4.toDF(schema=header_list)
>>> 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
```

df.printSchema()

```
>>> df.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)

>>>
```

Observations:

1) looking at the csv file provided, we found that the data is not proper structured format. We found that there are double quotes within each text and a semi-colon as delimiter
2) before creating ad data frame, we first to create rrds and clean up the data format
3) after the loading the data into a data frame and checking the schema we see that some numeric columns but the data types is appearing as string. We have to change the data types for these columns.

Data Frame Schema:

#Chaning the datatypes from string to Integer/double for numeric columns:

```
df1= df.withColumn("age", df['age'].cast('Integer')) \
    .withColumn("balance", df['balance'].cast('Double')) \
    .withColumn("day", df['day'].cast('Integer')) \
    .withColumn("duration", df['duration'].cast('Integer')) \
    .withColumn("campaign", df['campaign'].cast('Integer')) \
    .withColumn("pdays", df['pdays'].cast('Integer')) \
    .withColumn("previous", df['previous'].cast('Integer'))

df1.printSchema()
```

```
Traceback (most recent call last):
>>> df1= df.withColumn("age", df['age'].cast('Integer')) \
...         .withColumn("balance", df['balance'].cast('Double')) \
...         .withColumn("day", df['day'].cast('Integer')) \
...         .withColumn("duration", df['duration'].cast('Integer')) \
...         .withColumn("campaign", df['campaign'].cast('Integer')) \
...         .withColumn("pdays", df['pdays'].cast('Integer')) \
...         .withColumn("previous", df['previous'].cast('Integer'))
>>>
>>> df1.printSchema()
root
 |-- age: integer (nullable = true)
 |-- job: string (nullable = true)
 |-- marital: string (nullable = true)
 |-- education: string (nullable = true)
 |-- default: string (nullable = true)
 |-- balance: double (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)
```

Observations: The data types for the numeric columns are now changed correctly.

 # Writing the data frame with correct schema into a new csv file in HDFS

df1.write.option("header",True).option("inferSchema", "true").csv("/user/sujitsonargmail/spark_project/market_analysis_data_1.csv")

```
[sujitsonargmail@ip-10-0-41-79 ~]$ hadoop fs -ls spark_project
Found 2 items
-rw-r--r--   3 sujitsonargmail hadoop    5650234 2022-03-28 04:30 spark_project/market_analysis_data.csv
drwxr-xr-x   - sujitsonargmail hadoop          0 2022-04-01 06:16 spark_project/market_analysis_data_1.csv
[sujitsonargmail@ip-10-0-41-79 ~]$
```

#Reading the csv file for data analysis:

df = spark.read.option("header",True).option("inferSchema", "true").csv("/user/sujitsonargmail/spark_project/market_analysis_data_1.csv")
df.printSchema()

```
>>> df = spark.read.option("header",True).option("inferSchema", "true").csv("/user/sujitsonargmail/spark_project/market_analysis_data_1.csv")
>>> df.printSchema()
root
 |-- age: integer (nullable = true)
 |-- job: string (nullable = true)
 |-- marital: string (nullable = true)
 |-- education: string (nullable = true)
 |-- default: string (nullable = true)
 |-- balance: double (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)
```

Data

Data Exploration:

 # Checking any null values:

>>> df.describe(['age','balance','day','duration','campaign','pdays','previous']).show()

```
>>> df.describe(['job','marital','education','default','housing','loan','contact']).show()
+-------+-------+-------+---------+---------+-------+-------+-----+---------+
|summary|    job|marital|education|default|housing| loan| contact|
+-------+-------+-------+---------+---------+-------+-------+-----+---------+
|  count|  45211|  45211|    45211|  45211|  45211|45211|   45211|
|   mean|   null|   null|     null|   null|   null| null|    null|
| stddev|   null|   null|     null|   null|   null| null|    null|
|    min| admin.|divorced|  primary|     no|     no|   no|cellular|
|    max|unknown| single|  unknown|    yes|    yes|  yes| unknown|
+-------+-------+-------+---------+---------+-------+-------+-----+---------+
```

>>> df.describe(['job','marital','education','default','housing','loan','contact','month','poutcome','y']).show()

```
>>> df.describe(['job','marital','education','default','housing','loan','contact','month','poutcome','y']).show()
+-------+-------+-------+---------+---------+-------+-------+-----+--------+-----+--------+-----+
|summary|    job|marital|education|default|housing| loan| contact|month|poutcome|    y|
+-------+-------+-------+---------+---------+-------+-------+-----+--------+-----+--------+-----+
|  count|  45211|  45211|    45211|  45211|  45211|45211|   45211|45211|   45211|45211|
|   mean|   null|   null|     null|   null|   null| null|    null| null|    null| null|
| stddev|   null|   null|     null|   null|   null| null|    null| null|    null| null|
|    min| admin.|divorced|  primary|     no|     no|   no|cellular|  apr| failure|   no|
|    max|unknown| single|  unknown|    yes|    yes|  yes| unknown|  sep| unknown|  yes|
+-------+-------+-------+---------+---------+-------+-------+-----+--------+-----+--------+-----+
```

```
>>> df1 = df['age','balance','day','duration','campaign','pdays','previous']
>>> df1.summary().show()
```

```
>>> df1 = df['age','balance','day','duration','campaign','pdays','previous']
>>> df1.summary().show()
+-------+------------------+-----------------+-----------------+-----------------+-----------------+-----------------+-------------------+
|summary|               age|          balance|              day|         duration|         campaign|            pdays|           previous|
+-------+------------------+-----------------+-----------------+-----------------+-----------------+-----------------+-------------------+
|  count|             45211|            45211|            45211|            45211|            45211|            45211|              45211|
|   mean|40.936210214328374|1362.2720576850766|15.806418791886931|258.16307978146914| 2.763840658246887| 40.19782796222158|0.5803233726305546|
| stddev|10.618762040975421| 3044.765829168523|8.322476153044596|257.52781226517174|3.0980208832791707|100.12874599059819|2.3034410449312293|
|    min|                18|          -8019.0|                1|                0|                1|               -1|                  0|
|    25%|                33|             72.0|                8|              103|                1|               -1|                  0|
|    50%|                39|            448.0|               16|              180|                2|               -1|                  0|
|    75%|                48|           1427.0|               21|              319|                3|               -1|                  0|
|    max|                95|         102127.0|               31|             4918|               63|              871|                275|
+-------+------------------+-----------------+-----------------+-----------------+-----------------+-----------------+-------------------+
```

```
>>> df2 = df['job','marital','education','default','housing','loan','contact','month','poutcome','y']
>>> df2.summary().show()
```

```
>>> df2 = df['job','marital','education','default','housing','loan','contact','month','poutcome','y']
>>> df2.summary().show()
+-------+-------+--------+---------+-------+-------+-----+--------+-----+--------+-----+
|summary|    job| marital|education|default|housing| loan| contact|month|poutcome|    y|
+-------+-------+--------+---------+-------+-------+-----+--------+-----+--------+-----+
|  count|  45211|   45211|    45211|  45211|  45211|45211|   45211|45211|   45211|45211|
|   mean|   null|    null|     null|   null|   null| null|    null| null|    null| null|
| stddev|   null|    null|     null|   null|   null| null|    null| null|    null| null|
|    min| admin.|divorced|  primary|     no|     no|   no|cellular|  apr| failure|   no|
|    25%|   null|    null|     null|   null|   null| null|    null| null|    null| null|
|    50%|   null|    null|     null|   null|   null| null|    null| null|    null| null|
|    75%|   null|    null|     null|   null|   null| null|    null| null|    null| null|
|    max|unknown|  single|  unknown|    yes|    yes|  yes| unknown|  sep| unknown|  yes|
+-------+-------+--------+---------+-------+-------+-----+--------+-----+--------+-----+
```

```
from pyspark.sql.functions import col,isnan,when,count
df3 = df.select([count(when(col(c).contains('None') | \
                    col(c).contains('NULL') | \
                    (col(c) == '' ) | \
                    col(c).isNull() | \
                    isnan(c), c
                   )).alias(c)
                for c in df.columns])
df3.show()
```

```
>>> from pyspark.sql.functions import col,isnan,when,count
>>> df3 = df.select([count(when(col(c).contains('None') | \
...                             col(c).contains('NULL') | \
...                             (col(c) == '' ) | \
...                             col(c).isNull() | \
...                             isnan(c), c
...                            )).alias(c)
...                     for c in df.columns])
>>> df3.show()
+---+---+-------+---------+-------+-------+-------+----+-------+---+-----+--------+--------+-----+--------+--------+---+
|age|job|marital|education|default|balance|housing|loan|contact|day|month|duration|campaign|pdays|previous|poutcome|  y|
+---+---+-------+---------+-------+-------+-------+----+-------+---+-----+--------+--------+-----+--------+--------+---+
|  0|  0|      0|        0|      0|      0|      0|   0|      0|  0|    0|       0|       0|    0|       0|       0|  0|
+---+---+-------+---------+-------+-------+-------+----+-------+---+-----+--------+--------+-----+--------+--------+---+
```

Observations: There are no null or missing values in the data frame

Q2) a) Give marketing success rate (No. of people subscribed / total no. of entries)
b) Give marketing failure rate

#Checking total rows and columns in the df

>>> df = spark.read.option("header",True).option("inferSchema", "true").csv("/user/sujitsonargmail/spark_project/market_analysis_data_1.csv")>>> from pyspark.sql import functions as F >>> df.count(),len(df.columns)

```
>>> df = spark.read.option("header",True).option("inferSchema", "true").csv("/user/sujitsonargmail/spark_project/market_analysis_data_1.csv")
>>> from pyspark.sql import functions as F
>>> df.count(),len(df.columns)
(45211, 17)
>>>
```

a) Give marketing success rate

>>> df.count(),df.filter(F.col("y") =='yes').count()

```
>>> df.count(),df.filter(F.col("y") =='yes').count()
(45211, 5289)
>>>
```

>>> success_rate = round(df.filter(F.col("y")=='yes').count()/float(df.count()),2)
>>> success_rate

```
>>> success_rate = round(df.filter(F.col("y") =='yes').count()/float(df.count()),2)
>>> success_rate
0.12
```

Observations: As per the dataset only around 12 % of the targeted customers have the subscription *to term deposit*

b) Give marketing failure rate

>>> failure_rate = 1-success_rate
>>> failure_rate

```
>>> failure_rate = 1-success_rate
>>> failure_rate
0.88
```

Observations: around 88 % of the targeted customers have not subscribe *to term deposit*

Q3) Give the maximum, mean, and minimum age of the average targeted customer

>>> df.select('age').summary().show()

```
>>> df.select('age').summary().show()
+-------+-----------------+
|summary|              age|
+-------+-----------------+
|  count|            45211|
|   mean|40.936210214328 37|
| stddev|10.618762040975 42|
|    min|               18|
|    25%|               33|
|    50%|               39|
|    75%|               48|
|    max|               95|
+-------+-----------------+
```

Observations: average age of the targeted customers is around 40 yrs with 95 being the max age and 18 being the min age. This shows that most of the customers are in an age group of 35-40

Q4) Check the quality of customers by checking average balance, median balance of customers
df.select('balance').summary().show()

```
>>> df.select('balance').summary().show()
+-------+------------------+
|summary|           balance|
+-------+------------------+
|  count|             45211|
|   mean|1362.2720576850766|
| stddev| 3044.765829168523|
|    min|           -8019.0|
|    25%|              72.0|
|    50%|             448.0|
|    75%|            1427.0|
|    max|          102127.0|
+-------+------------------+
```

```
>>> from pyspark.sql.functions import *
>>> from pyspark.sql.functions import when
>>> df2 = df.withColumn("balance_group", when(df.balance <= -1000,"Negative Balance 1k and above")
                                        .when(df.balance <= -500,"Negative Balance 500 to 1K")
                                        .when(df.balance <= -100,"Negative Balance 100 to 500")
                                        .when(df.balance <= -1,"Negative Balance 1 to 100")
                                        .when(df.balance == 0,"Zero balance")
                    .when((df.balance > 0) & (df.balance < 100),"below 100")
                    .when((df.balance >=100) & (df.balance < 500),"below 500")
                    .when((df.balance >=500) & (df.balance < 100),"below 1000")
                    .when((df.balance >=100) & (df.balance < 10000),"below 10000")
                    .otherwise("Obove 10k"))

>>> df3 = df2.withColumn("age_group", when(df2.age < 26,"Young")
                    . when((df2.age > 26) & (df2.age < 51),"Mid")
                    .otherwise("Old"))
```

 # Looking at the balance of the customers who has subscribed to the term deposit

>>> df4 = df3.filter(df3.y =='yes').groupby('age_group','marital','balance_group').count()
*>>> from pyspark.sql.functions import **
>>> df4.sort(desc('count')).show()

```
>>> from pyspark.sql.functions import *
>>> df4.sort(desc('count')).show()
+---------+--------+-------------+-----+
|age_group| marital|balance_group|count|
+---------+--------+-------------+-----+
|      Mid| married|  below 10000|  963|
|      Mid|  single|  below 10000|  781|
|      Old| married|  below 10000|  655|
|      Mid| married|    below 500|  418|
|      Mid|  single|    below 500|  367|
|      Old| married|    below 500|  180|
|      Mid|divorced|  below 10000|  165|
|      Old|divorced|  below 10000|  151|
|    Young|  single|  below 10000|  144|
|      Mid| married|    below 100|  135|
|      Mid|  single|    below 100|  122|
|      Mid| married| Zero balance|  116|
|    Young|  single|    below 500|  113|
|      Old|  single|  below 10000|  101|
|      Mid|divorced|    below 500|   88|
|      Mid|  single| Zero balance|   58|
|      Old| married| Zero balance|   54|
|      Old| married|    below 100|   52|
|      Old|  single|    below 500|   48|
|      Old|divorced|    below 500|   47|
+---------+--------+-------------+-----+
only showing top 20 rows
```

>>> df4 = df3.filter(df3.y =='yes').groupby('balance_group').count()
>>> df4.sort(desc('count')).show(truncate=False)

```
>>> df4 = df3.filter(df3.y =='yes').groupby('balance_group').count()
>>> df4.sort(desc('count')).show(truncate=False)
+---------------------------+-----+
|balance_group              |count|
+---------------------------+-----+
|below 10000                |2972 |
|below 500                  |1269 |
|below 100                  |411  |
|Zero balance               |292  |
|Obove 10k                  |135  |
|Negative Balance 100 to 500|112  |
|Negative Balance 1 to 100  |60   |
|Negative Balance 500 to 1K |33   |
|Negative Balance 1k and above|5    |
+---------------------------+-----+
```

 # Looking at the balance of the customers who has not yet subscribed to the term deposit

>>> df4 = df3.filter(df3.y =='no').groupby('age_group','marital','balance_group').count()

*>>> from pyspark.sql.functions import **

>>> df4.sort(desc('count')).show()

```
>>> df4 = df3.filter(df3.y =='no').groupby('age_group','marital','balance_group').count()
>>> df4.sort(desc('count')).show()
+---------+--------+-------------------+-----+
|age_group| marital|       balance_group|count|
+---------+--------+-------------------+-----+
|      Mid| married|         below 10000| 7924|
|      Mid| married|           below 500| 4529|
|      Mid|  single|         below 10000| 3972|
|      Old| married|         below 10000| 3191|
|      Mid|  single|           below 500| 2444|
|      Mid| married|           below 100| 2075|
|      Mid| married|        Zero balance| 1544|
|      Old| married|           below 500| 1330|
|      Mid|divorced|         below 10000| 1193|
|      Mid|  single|           below 100| 1164|
|      Mid| married|Negative Balance ...|  922|
|      Mid|divorced|           below 500|  796|
|      Old|divorced|         below 10000|  634|
|      Mid|  single|        Zero balance|  606|
|      Old| married|           below 100|  593|
|      Old| married|        Zero balance|  546|
|      Old|  single|         below 10000|  459|
|      Mid|divorced|           below 100|  442|
|      Mid| married|Negative Balance ...|  439|
|      Mid|  single|Negative Balance ...|  429|
+---------+--------+-------------------+-----+
only showing top 20 rows
```

>>> df4 = df3.filter(df3.y =='no').groupby('balance_group').count()

>>> df4.sort(desc('count')).show(truncate=False)

```
>>> df4 = df3.filter(df3.y =='no').groupby('balance_group').count()
>>> df4.sort(desc('count')).show(truncate=False)
+-----------------------------+-----+
|balance_group                |count|
+-----------------------------+-----+
|below 10000                  |17773|
|below 500                    |9931 |
|below 100                    |4746 |
|Zero balance                 |3222 |
|Negative Balance 100 to 500  |1949 |
|Negative Balance 1 to 100    |949  |
|Obove 10k                    |694  |
|Negative Balance 500 to 1K   |519  |
|Negative Balance 1k and above|139  |
+-----------------------------+-----+
```

Observations:

1)The average balance of the customers is around 1362 whereas the median value is 448. The max value is 102127 which appears to be an outlier.

2) most of the customers who has subscribed has balance 100 to 10000

3) by checking the customers by age group and balance group and marital status, we see that most of customers who has the subscription are at mid age group who are married/single and has balance below 10k

4) the same trend is with the customers who has not done the subscription yet where we see the customers to are married/single, mid age group and balance below 10k.

Q5) Check if age matters in marketing subscription for deposit

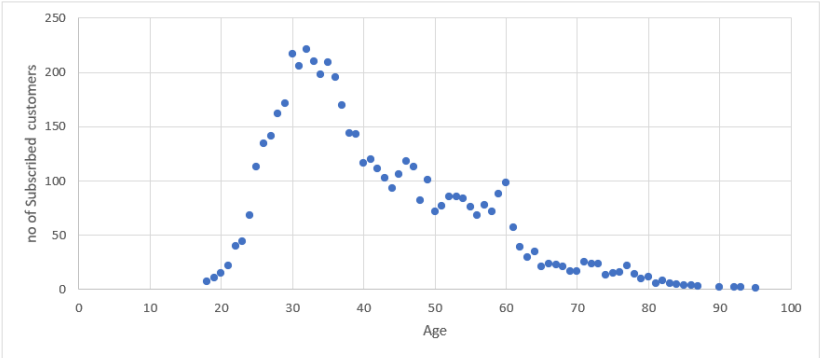
>>> agedf = df.filter(df.y =='yes').groupby('age').count()

>>> agedf.show()

```
>>> agedf=df.filter(df.y=='yes').groupby('age').count()
>>> agedf.show()
+---+-----+
|age|count|
+---+-----+
| 31|  206|
| 85|    4|
| 65|   21|
| 53|   85|
| 78|   14|
| 34|  198|
| 81|    6|
| 28|  162|
| 76|   16|
| 26|  134|
```

>>> agedf .sort(desc('count')).show(100)

```
>>> agedf.sort(desc('count')).show(100)
+---+-----+
|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|
| 26|  134|
| 41|  120|
| 46|  118|
| 40|  116|
| 25|  113|
| 47|  113|
| 42|  111|
| 45|  106|
| 43|  103|
| 49|  101|
| 60|   98|
| 44|   93|
```



Observations: Most of the customers who has subscribed are at the age group of 30 to 40

Q6) Check if marital status mattered for a subscription to deposit

>>> df.filter(df.y=='yes').groupby('marital').count().show()

```
>>> df.filter(df.y=='yes').groupby('marital').count().show()
+--------+-----+
| marital|count|
+--------+-----+
|divorced|  622|
| married| 2755|
|  single| 1912|
+--------+-----+
```

Observations: Married customers has the highest subscription followed by single customers.

Q7) Check if age and marital status together mattered for a subscription to deposit scheme

>>> agemaritaldf=df.filter(df.y=='yes').groupby('age','marital').count()
>>> agemaritaldf.show()

*>>> from pyspark.sql.functions import **
>>> agemaritaldf.sort(desc('count')).show()

```
>>> agemaritaldf.sort(desc('count')).show()
+---+-------+-----+
|age|marital|count|
+---+-------+-----+
| 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|
| 32|married|   87|
| 39|married|   87|
| 38|married|   86|
| 35| single|   84|
| 47|married|   83|
| 31|married|   80|
+---+-------+-----+
only showing top 20 rows
```

Observations: Single person at age group of 30 to 36 shows most subscription

Q8) Do feature engineering for the bank and find the right age effect on the campaign.

```
from pyspark.sql.functions import when
df2 = df.withColumn("age_group", when(df.age < 26,"Young")
                    . when((df.age > 26) & (df.age < 51),"Mid")
                    .otherwise("Old"))
df2.show()
```

```
>>> from pyspark.sql.functions import when
>>> df2 = df.withColumn("age_group", when(df.age < 26,"Young")
...                                 . when((df.age > 26) & (df.age < 51),"Mid")
...                                 .otherwise("Old"))
>>> df2.show()
+---+-------------+--------+---------+-------+-------+-------+----+--------+---+-----+--------+--------+-----+--------+--------+---+---------+
|age|          job| marital|education|default|balance|housing|loan| contact|day|month|duration|campaign|pdays|previous|poutcome|  y|age_group|
+---+-------------+--------+---------+-------+-------+-------+----+--------+---+-----+--------+--------+-----+--------+--------+---+---------+
| 52|     services| married|secondary|     no|  381.0|     no| yes|cellular| 25|  aug|     288|      12|   -1|       0| unknown| no|      Old|
| 53|      unknown| married|  unknown|     no|    0.0|     no|  no|cellular| 25|  aug|     209|       5|   -1|       0| unknown| no|      Old|
| 51|   technician| married| tertiary|     no|   -3.0|     no|  no|cellular| 25|  aug|      91|       9|   -1|       0| unknown| no|      Old|
| 33|   technician|  single|secondary|     no|  -32.0|     no|  no|cellular| 25|  aug|     196|      12|   -1|       0| unknown| no|      Mid|
| 48|   management|divorced| tertiary|     no|    0.0|     no|  no|cellular| 25|  aug|     110|       3|   -1|       0| unknown| no|      Mid|
| 60|      retired| married|  primary|     no|  155.0|     no|  no|cellular| 25|  aug|     115|       7|   -1|       0| unknown| no|      Old|
| 50|   management|divorced| tertiary|     no|    0.0|     no|  no|cellular| 25|  aug|      57|       3|   -1|       0| unknown| no|      Mid|
| 59|  blue-collar| married|  primary|     no| 6271.0|    yes|  no|cellular| 25|  aug|     102|       5|   -1|       0| unknown| no|      Old|
| 33|   technician|  single| tertiary|     no|  137.0|     no|  no|cellular| 25|  aug|      88|       4|   -1|       0| unknown| no|      Mid|
| 37|self-employed| married|secondary|     no|  119.0|     no|  no|cellular| 25|  aug|      68|       4|   -1|       0| unknown| no|      Mid|
| 45|  blue-collar| married|  primary|     no|  185.0|     no|  no|cellular| 25|  aug|      78|       4|   -1|       0| unknown| no|      Mid|
| 47|   management| married|secondary|     no| 1083.0|     no|  no|cellular| 25|  aug|     141|       4|   -1|       0| unknown| no|      Mid|
| 41|   technician| married|secondary|     no| 2039.0|     no|  no|cellular| 25|  aug|     160|       4|   -1|       0| unknown| no|      Mid|
| 52|   management| married| tertiary|     no|  967.0|     no|  no|cellular| 25|  aug|     472|      10|   -1|       0| unknown| no|      Old|
| 35|   technician|  single| tertiary|     no|  275.0|    yes|  no|cellular| 25|  aug|      63|       5|   -1|       0| unknown| no|      Mid|
| 34|   technician| married|secondary|     no|   47.0|     no|  no|cellular| 25|  aug|     132|       6|   -1|       0| unknown| no|      Mid|
| 36|   management| married| tertiary|     no| 1235.0|     no|  no|cellular| 25|  aug|      85|       6|   -1|       0| unknown| no|      Mid|
| 32|   technician| married|secondary|    yes|    4.0|    yes| yes|cellular| 25|  aug|     132|       8|   -1|       0| unknown| no|      Mid|
| 36|   management| married| tertiary|     no| 3874.0|     no|  no|cellular| 25|  aug|     425|       6|   -1|       0| unknown| no|      Mid|
| 58|  blue-collar| married|  unknown|     no|    9.0|     no|  no|cellular| 25|  aug|      50|      23|   -1|       0| unknown| no|      Old|
+---+-------------+--------+---------+-------+-------+-------+----+--------+---+-----+--------+--------+-----+--------+--------+---+---------+
only showing top 20 rows
```

```
>>> df2.filter(df2.y=='yes').groupby('age_group').count().show()
```

```
>>> df2.filter(df2.y=='yes').groupby('age_group').count().show()
+---------+-----+
|age_group|count|
+---------+-----+
|      Old| 1447|
|    Young|  320|
|      Mid| 3522|
+---------+-----+
```

Observations: Mid aged are the most subscribed customer group in the given dataset