**Market Analysis in Banking Domain**

**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.

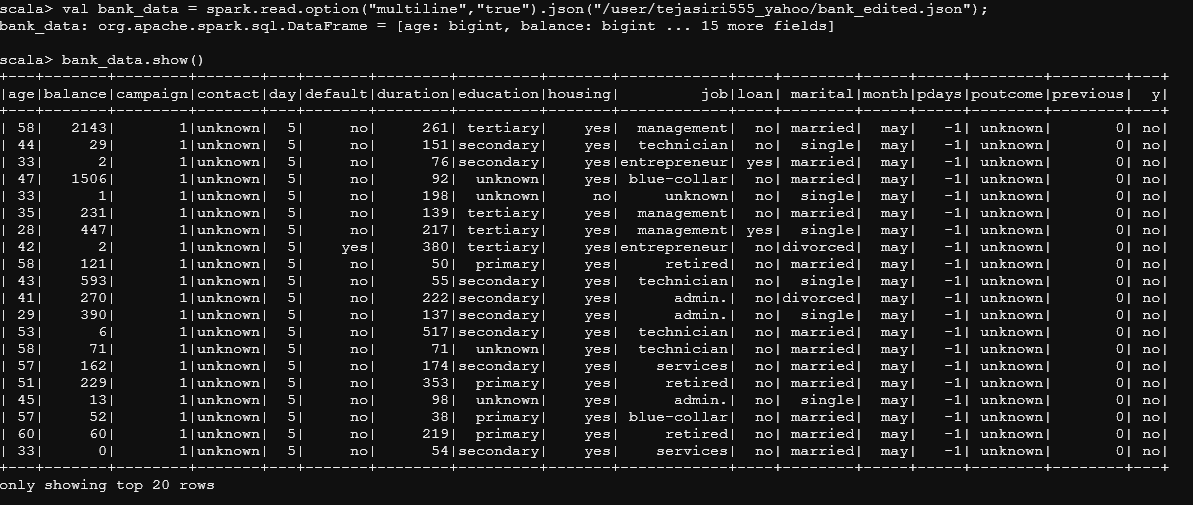
**Datasets:**

bank\_edited.json

**Source code:**

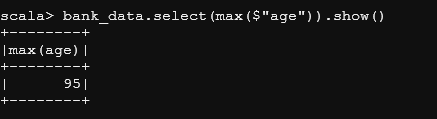
* From web console enter into spark-scala shell by typing spark-shell.

Load data and create a Spark data frame

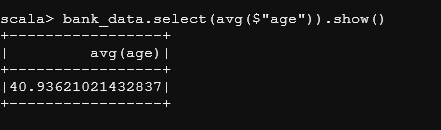
* import scala.reflect.runtime.universe
* import org.apache.spark.SparkConf
* import org.apache.spark.SparkContext
* import org.apache.spark.ml.Pipeline
* import org.apache.spark.ml.classification.LogisticRegression
* import org.apache.spark.ml.feature.Bucketizer
* import org.apache.spark.ml.feature.Normalizer
* import org.apache.spark.ml.feature.StringIndexer
* import org.apache.spark.ml.feature.VectorAssembler
* import org.apache.spark.mllib.evaluation.BinaryClassificationMetrics
* import org.apache.spark.sql.DataFrame
* import org.apache.spark.sql.SQLContext
* import org.apache.spark.sql.functions.mean
* val bank\_data = spark.read.option("multiline","true").json("/user/tejasiri555\_yahoo/bank\_edited.json")
* **Output:**
* bank\_data.registerTempTable("datatable")

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

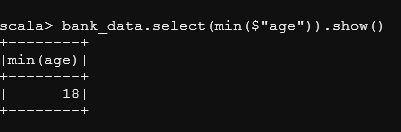
* bank\_data.select(max($"age")).show()
* **Output:**



* bank\_data.select(avg($"age")).show()
* **Output:**

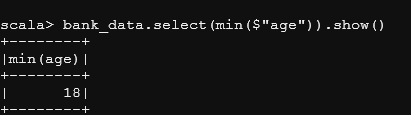


* bank\_data.select(min($"age")).show()
* **Output:**

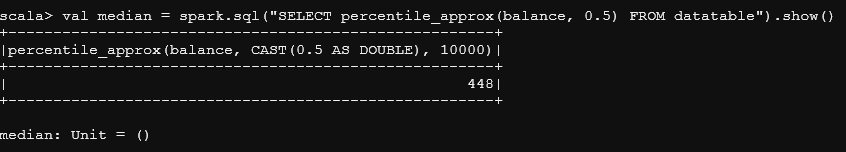


Check the quality of customers by checking average balance, median balance of customers

* bank\_data.select(avg($"balance")).show()
* **Output:**

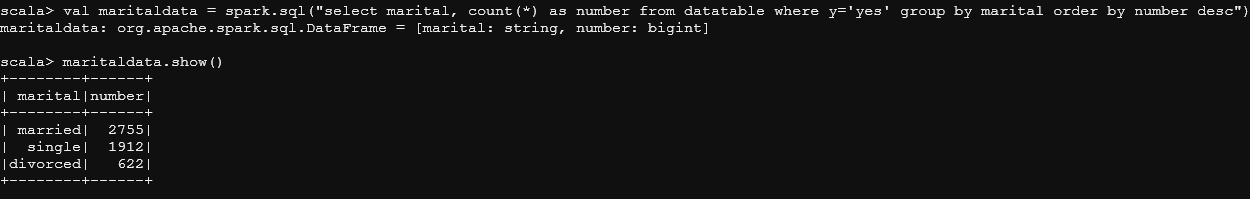


* val median = spark.sql("SELECT percentile\_approx(balance, 0.5) FROM datatable").show()
* **Output:**



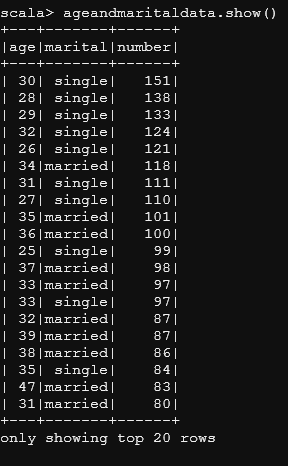
Check if age matters in marketing subscription for deposit

* val maritaldata = spark.sql("select marital, count(\*) as number from datatable where y='yes' group by marital order by number desc")
* maritaldata.show()
* **Output**:



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

* val ageandmaritaldata = spark.sql("select age, marital, count(\*) as number from datatable where y='yes' group by age,marital order by number desc")
* ageandmaritaldata.show()
* **Output:**



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

val agedata = spark.udf.register("agedata",(age:Int) => {

if (age < 20)

"Teen"

else if (age > 20 && age <= 32)

"Young"

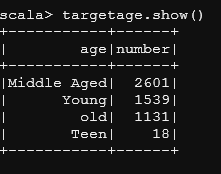
else if (age > 33 && age <= 55)

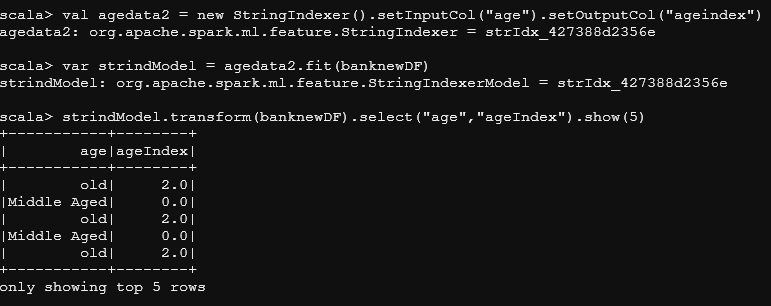
"Middle Aged"

else

"old"

})

* **Insight:** Funtion is defined for age specification.
* val banknewDF = bank\_data.withColumn("age",agedata(bank\_data("age")))
* banknewDF.show()
* **Insight**: Replacing the old age column with the new age column.
* **Output:**
* banknewDF.registerTempTable("banknewtable")
* val targetage = spark.sql("select age, count(\*) as number from banknewtable where y='yes' group by age order by number desc")
* targetage.show()
* **Insight:** Checking which age group subscribed the most.
* **Output:**
* val agedata2 = new StringIndexer().setInputCol("age").setOutputCol("ageindex")
* **Insight:** pipelining with string Indexer.
* var strindModel = agedata2.fit(banknewDF)
* **Insight:** Fitting the model
* strindModel.transform(banknewDF).select("age","ageIndex").show(5)
* **Insights:** assigns generated value of index of the column, by feature engineering.
  + Middle aged is the most frequent word in this data, so it is given as “0”.
* **Output:**



* **Insights:** We can conclude from Feature Engineering that it is the “Middle aged” people between age 33 and 55 who should be targeted customers as they subscribe the most.