**Market Analysis in Banking Domain**

**1. Load data and create Spark data frame**

import org.apache.spark.sql.SQLContext

val sqlContext = new SQLContext(sc)

val dfs = sqlContext.read.format("com.databricks.spark.csv").option("header","true").option("inferSchema","true").option("delimiter",";").load("/user/yenugusivakumargmail/Inputfiles/bankdata.csv")

dfs.registerTempTable("bank")

sqlContext.sql("""select \* from bank limit 10""").show();

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

val total\_count=dfs.count()

val yes\_count=dfs.filter("y='yes'").count()

val success\_rate = yes\_count/total\_count.toFloat\*100

val no\_count=dfs.filter("y='no'").count()

val fail\_rate = no\_count.toFloat/total\_count\*100

**3. Maximum, Mean, and Minimum age of average targeted customer**

val age\_stats = sqlContext.sql("select max(age), avg(age), min(age) from bank")

age\_stats.show()

**4. Check quality of customers by checking average balance, median balance of customers**

val customer\_quality = sqlContext.sql("select avg(balance) AS Average\_balance, percentile\_approx(balance, .5) as Median\_balance from bank")

customer\_quality.show()

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

val age\_mktg=sqlContext.sql("select y, avg(age) from bank group by y")

age\_mktg.show()

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

val marital\_mktg=sqlContext.sql("select marital, y, count(marital) from bank group by marital, y order by y")

marital\_mktg.show()

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

val age\_mktg\_subs=sqlContext.sql("select marital, y, count(marital),avg(age) from bank group by marital, y order by y")

age\_mktg\_subs.show()

**8. Do feature engineering for column—age and find right age effect on campaign**

val age\_camp = sqlContext.sql("select y, age, count(y) from bank group by y, age order by y desc, age")

age\_camp.show()

val agecat=sqlContext.sql("select \*, case when age<30 then '18-30' else (case when age>30 and age<60 then '31-60' else '>65' end)end as age\_category from bank order by age asc")

agecat.registerTempTable("bank2")

sqlContext.sql("select age\_category, y,count(\*) from bank2 group by age\_category, y order by y").show()