**pyspark-sql-assignments**

Assignment #2

Bank Marketing Campaign Data Analysis with DataFrame API

a) Load Bank Marketing Dataset and create DataFrame

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

c) Give marketing failure rate

d) Maximum, Mean, and Minimum age of the average targeted customer

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

f) Check if age matters in marketing subscription for deposit

g) Show AgeGroup [Teenagers, Youngsters, MiddleAgers, Seniors] wise Subscription Count.

g) Check if marital status mattered for subscription to deposit.

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

* Created a bankingmarket.py file

from pyspark.sql import SparkSession

from pyspark.sql.functions import \*

spark = SparkSession.builder.appName('prac05').getOrCreate()

spark

df\_b = spark.read.option("delimiter", ";").option("header", True).option('inferSchema', True).csv("/home/mkm/futurense\_hadoop-pyspark/labs/dataset/bankmarket/bankmarketdata.csv")

df\_b.printSchema()

df\_b.show(5)

df\_b.registerTempTable("bankings")

q = '''

select

case

when age < 20 then 'Under 20'

when age between 21 and 30 then '21-30'

when age between 31 and 40 then '31-40'

when age between 41 and 50 then '41-50'

when age between 51 and 60 then '51-60'

when age > 60 then 'Over 60'

END as age\_range,

COUNT(IF(y = 'yes', 1, NULL)) as no\_of\_deposits,

count(\*) as tot\_no\_cust,

round( (COUNT(IF(y = 'yes', 1, NULL)) / count(\*)) \* 100, 2) as succ\_rate

from bankings

group by age\_range

'''

df\_ag = spark.sql(q)

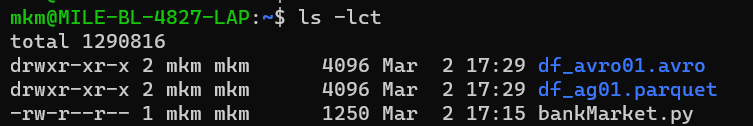
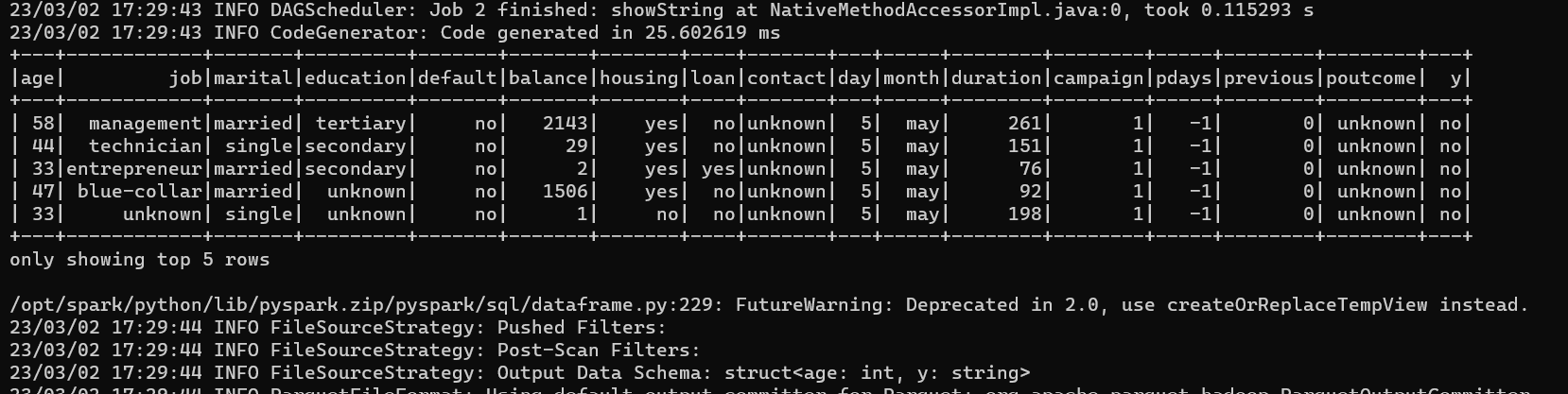
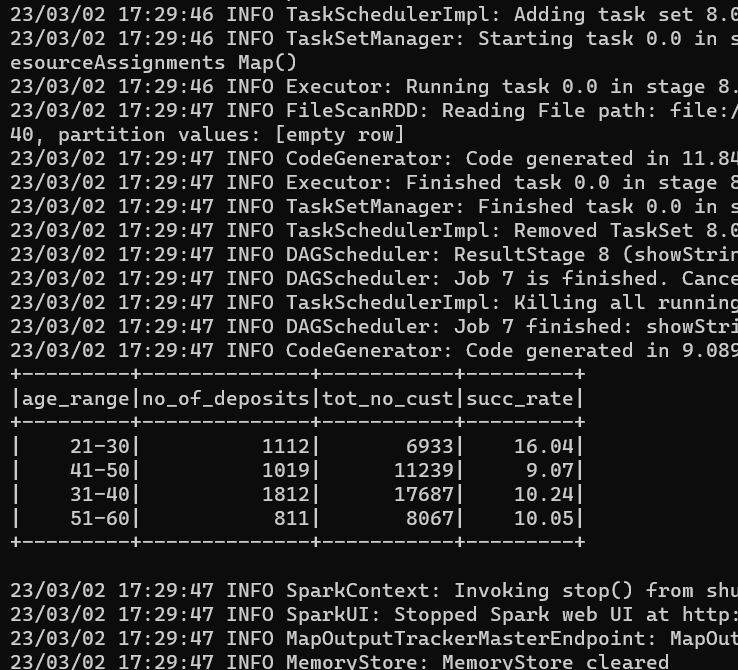
df\_ag.select(\*df\_ag.columns).write.format("parquet").save("/home/mkm/df\_ag01.parquet")

df\_read = spark.read.format("parquet").load("/home/mkm/df\_ag01.parquet")

df\_read = df\_read.filter(df\_read.tot\_no\_cust > 2000)

df\_read.select(\*df\_read.columns).write.format("avro").save("/home/mkm/df\_avro01.avro")

spark.read.format("avro").load("/home/mkm/df\_avro01.avro").show()

* 
* Command to run in local mode
* spark-submit --packages org.apache.spark:spark-avro\_2.12:3.3.2 bankMarket.py
* command to run in cluster mode
* spark-submit --packages org.apache.spark:spark-avro\_2.12:3.3.2 bankMarket.py --master spark://MILE-BL-4827-LAP.:7077
* 
* 
* Creating a cronjob
* crontab -e
* \* \* \* \* \* spark-submit --packages org.apache.spark:spark-avro\_2.12:3.3.2 bankMarket.py
* This will run job for every minute

‘’’

\* \* \* \* \* command to be executed

- - - - -

| | | | |

| | | | ----- Day of week (0 - 7) (Sunday=0 or 7)

| | | ------- Month (1 - 12)

| | --------- Day of month (1 - 31)

| ----------- Hour (0 - 23)

------------- Minute (0 - 59)

‘’’