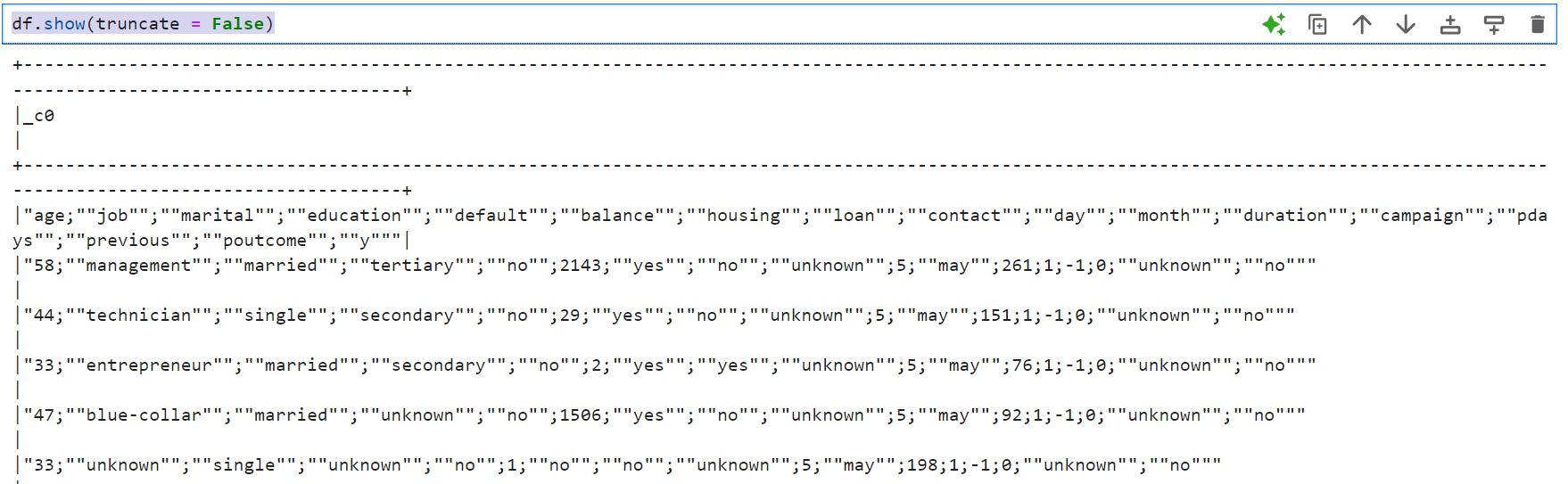
**Project – Marketing Analysis**

**Question 1: Load data and create Spark data frame**

1. df = spark.read.format("csv").option("header", False).option("delimiter","\t").load(r"C:\Users\venka\Desktop\BigDataSL\BD-C-2\Project1\_MarketingAnalysis\Project 1\_dataset\_bank-full.csv")
2. df.show(truncate = False)



Note: Further engineering is required to make the df clean and well-structured.

1. df.select( regexp\_replace(col("\_c0"),"\"", "").alias("nc")).coalesce(1).write.format("csv").option("delimiter", "|").option("header",False).save(r"C:\Users\venka\Desktop\BigDataSL\BD-C-2\Project1\_MarketingAnalysis\Modified\_Data")
2. df = spark.read.format("csv").option("header", True).option("delimiter", ";").option("inferSchema", True).load(r"C:\Users\venka\Desktop\BigDataSL\BD-C-2\Project1\_MarketingAnalysis\Modified\_Data\part-00000-b1cfbee6-3b53-4e4b-888a-c7bff667c0a0-c000.csv")
3. df.show(truncate = False)

A screen shot of a computer

Description automatically generated

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

1. total\_entries = df.count()
2. total\_entries

A screen shot of a computer

Description automatically generated

1. subscribed\_entries = df.filter(col("y")=='yes').count()

A close-up of a computer screen

Description automatically generated

1. success\_rate = subscribed\_entries / total\_entries

A screenshot of a computer

Description automatically generated

1. success\_percent = (success\_rate )\*100

A screenshot of a computer code

Description automatically generated

**Question 2a: Give marketing failure rate**

1. failure\_rate = (1-success\_rate)

A screenshot of a computer

Description automatically generated

1. failure\_percent = (failure\_rate)\*100

A screenshot of a computer

Description automatically generated

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

1. df.agg(max(col("age")).alias("max\_age")).show()

A computer screen with text

Description automatically generated

1. df.agg(round(mean(col("age")), 2).alias("avg\_age")).show()

A screen shot of a computer

Description automatically generated

1. df.agg(min(col("age")).alias("min\_age")).show()

A computer screen with text

Description automatically generated

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

1. df.groupBy(col("y")).agg(avg(col("balance")).alias("avg\_balance")).show()

A computer screen shot of a web browser

Description automatically generated

Inference: We may infer that, on average, customers who are likely to subscribe have a higher bank balance.

1. df.groupBy(col("y")).agg(median(col("balance")).alias("median\_balance")).show()

A computer screen with text on it

Description automatically generated

Inference: We may infer that 50% of subscribers have a balance higher than 733.0 while 50% of customers who haven’t subscribed have balance less than or equal to 417.0.

Note: Since median is not much affected by the outliers, we may stick to the inference that the customers who are likely to subscribe have a higher bank balance.

**Question 5: Check if age matters in marketing subscription for deposit**

1. df.groupBy(col("y")).agg(avg(col("age"))).show()

A computer screen shot of a computer code

Description automatically generated

Inference: We may infer that age may not matter in the subscription since there is no significant difference in the average age between the two sects.

1. df.groupBy(col("y")).agg(median(col("age"))).show()

A screenshot of a computer code

Description automatically generated

Note: We may stick to our inference that the age of the customer may not matter for them to subscribe.

**Question 6: Check if marital status mattered for subscription to deposit**

1. df.groupBy(col("marital")).agg(sum(when(col("y")=='yes',1).otherwise(0)).alias("yes\_count")).show()

A computer screen shot of a computer screen

Description automatically generated

1. df.groupBy(col("marital")).count().show()

A screenshot of a computer code

Description automatically generated

Inference: 14.9% of single people, 11.9% of divorced people, and 10.1% of married people have subscribed. Considering that the percentages are not so significant, there is no proof to say that marital status matters for subscription.

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

1. df.groupBy([col("y"), col("marital")]).agg(mean("age").alias("avg\_age")).orderBy(col("marital")).show()

A computer screen shot of a computer

Description automatically generated

Inference: Based on the above analysis, we may infer that, for single people, age doesn’t matter for subscription. For married and divorced individuals, higher age may slightly favor the subscription.

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

1. featured\_df = df.withColumn("age\_bracket", expr("CASE WHEN age > 55 THEN 'Old' WHEN age > 33 THEN 'Middle' ELSE 'Young' END"))
2. featured\_df = featured\_df.groupBy([col("age\_bracket"), col("y")]).count()

A screenshot of a computer code

Description automatically generated

1. total\_count\_df = featured\_df.groupBy(col("age\_bracket")).agg(sum(col("count")).alias("total\_count"))

A screenshot of a computer code

Description automatically generated

1. join\_df = featured\_df.join(total\_count\_df, on="age\_bracket")

A screenshot of a computer code

Description automatically generated

1. join\_df = join\_df.withColumn("percent", round((col("count")/col("total\_count"))\*100, 2))

A screenshot of a computer

Description automatically generated

Inference: 9.5% of middle-aged people, 13.6% young people, and 18.5% of older people have subscribed. Since the percentages are not significant, it is not viable to state that age has an actual effect on this campaign.