**Coding Assignment**

Table of Contents

[Spark Batch 2](#_Toc87527084)

[1. Read attached json (demographic\_info.json) into a Spark Dataframe and carry out following 2](#_Toc87527085)

[Query: 2](#_Toc87527086)

[Screenshot: 2](#_Toc87527087)

[2. Use following apis over Dataframe select to show columns (name, age, gender, isActive, balance, company, eyeColor, email, phone) filter to show records with isActive as true 2](#_Toc87527088)

[Query 2](#_Toc87527089)

[Screenshot: 3](#_Toc87527090)

[3. Show top 2 male and female with maximum balance 3](#_Toc87527091)

[Query 3](#_Toc87527092)

[Screenshot 3](#_Toc87527093)

[4. Add a column Age\_group with classifications as Teenager (13-19 years), Young (20-40 years), Old (>40 years) 4](#_Toc87527094)

[Query 4](#_Toc87527095)

[Screenshot 4](#_Toc87527096)

[5. Create temp table view over initially read json Dataframe and run sql queries for same requirements given above. 5](#_Toc87527097)

[6. Convert above selected column dataframe into an RDD and save it into a text file. 7](#_Toc87527098)

[Query 7](#_Toc87527099)

[Screenshot 7](#_Toc87527100)

[Kafka 8](#_Toc87527101)

[Start Server 8](#_Toc87527102)

[1. Create a topic with 3 partitions 8](#_Toc87527103)

[Query 8](#_Toc87527104)

[Screenshot 8](#_Toc87527105)

[2. Create a producer writing data to above created kafka topic with following considerations 8](#_Toc87527106)

[Code 8](#_Toc87527107)

[Screenshot 9](#_Toc87527108)

[Structured Streaming 10](#_Toc87527109)

[1. Create a Spark streaming job reading data from above created kafka topic with following considerations 10](#_Toc87527110)

[Code 10](#_Toc87527111)

[Screenshot 10](#_Toc87527112)

For below mentioned exercise, share relevant code and snapshots

# Spark Batch

1. Read attached json (demographic\_info.json) into a Spark Dataframe and carry out following –

### Query:

spark.read.json("/Users/abhishekkashyap/Desktop/Big Data/demographic\_info.json")

### Screenshot:

Text

Description automatically generated

## Use following apis over Dataframe **select** to show columns (name, age, gender, isActive, balance, company, eyeColor, email, phone) **filter** to show records with isActive as true

Query:

res0.filter("isActive == 'true'").select("name", "age", "gender", "isActive", "balance", "company", "eyeColor", "email", "phone").show()

### Screenshot:

Graphical user interface, text

Description automatically generated

## Show top 2 male and female with maximum balance

Query:

import org.apache.spark.sql.expressions.Window

val win=Window.partitionBy("gender").orderBy(desc("balance"))

res0.withColumn("rank", rank().over(win).alias("rank")).filter(col("rank") <= 2).show()

Screenshot:

Graphical user interface

Description automatically generated

## Add a column Age\_group with classifications as Teenager (13-19 years), Young (20-40 years), Old (>40 years)

Query:

res0.withColumn("Age\_group", when($"age">=13 and $"age"<=19, "Teenager").otherwise(when($"age">=20 and $"age"<=40, "Young").otherwise("Old"))).show()

Screenshot:

res0.withColumn("Age\_group", when($"age">=13 and $"age"<=19, "Teenager").otherwise(when($"age">=20 and $"age"<=40, "Young").otherwise("Old"))).show()

Graphical user interface, text

Description automatically generated

## Create temp table view over initially read json Dataframe and run sql queries for same requirements given above.

Query:

res0.createOrReplaceTempView("temp")

spark.table("temp").filter("isActive == 'true'").select("name", "age", "gender", "isActive", "balance", "company", "eyeColor", "email", "phone").show()

Screenshot

Graphical user interface, text

Description automatically generated

val win=Window.partitionBy("gender").orderBy(desc("balance"))

spark.table("temp").withColumn("rank", rank().over(win).alias("rank")).filter(col("rank") <= 2).show()

A screenshot of a computer

Description automatically generated with medium confidence

spark.table("temp").withColumn("Age\_group", when($"age">=13 and $"age"<=19, "Teenager").otherwise(when($"age">=20 and $"age"<=40, "Young").otherwise("Old"))).show()

Graphical user interface, text

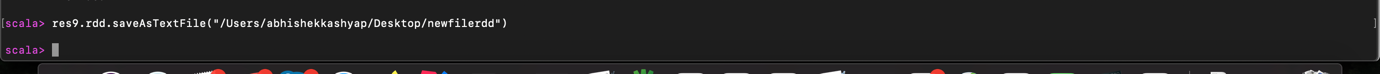
Description automatically generated

## Convert above selected column dataframe into an RDD and save it into a text file.

Query:

res9.rdd.saveAsTextFile("/Users/abhishekkashyap/Desktop/newfilerdd")

Screenshot:



Graphical user interface, application

Description automatically generated

# Kafka

## Start Server

zookeeper-server-start /usr/local/etc/kafka/zookeeper.properties & kafka-server-start /usr/local/etc/kafka/server.properties

## Create a topic with 3 partitions

Query:

kafka-topics --create --bootstrap-server localhost:9092 --partitions 3 --topic nagp-topic --replication-factor 1

Screenshot:



1. Create a producer writing data to above created kafka topic with following considerations –
   1. messages should be read from a file line by line (use any file from your side with limited content)
   2. message should be produced on kafka topic in (key, Value) format where key is timestamp+index and value is actual message

Code:  
NAGPProducer java file in attached Spark-Producer project.

Screenshot:

A screenshot of a computer

Description automatically generated with medium confidence

# Structured Streaming

1. Create a Spark streaming job reading data from above created kafka topic with following considerations –

* + - 1. read from beginning
      2. calculate word count
      3. print word count to console

Code:

StreamFromKafka java file in attached Spark-streaming project.

Screenshot:

A screenshot of a computer

Description automatically generated with medium confidence