Code Logic - Retail Data Analysis

Following are the steps followed in writing the code to read the data from Kafka topics

1.**Initiated Spark Session**

```
# initiating spark session
spark = SparkSession \
    .builder \
    .appName("retailstream") \
    .getOrCreate()
spark.sparkContext.setLogLevel('ERROR')
```

2. Read the data from the Kafka topic using readStream

```
# reading data of given topic from kafka bootstrap server
lines = spark \
    .readStream \
    .format("kafka") \
    .option("kafka.bootstrap.servers", "18.211.252.152:9092") \
    .option("subscribe", "real-time-project") \
    .option("failOnDataloss", "false") \
    .option("startingOffsets", "earliest") \
    .load()
```

3. Defined Schema

```
# Schema for data
myschema = StructType([
    StructField("country", StringType()),
    StructField("invoice_no", LongType()),
    StructField("items", ArrayType(
        StructType([
            StructField("SKU", StringType()),
            StructField("title", StringType()),
            StructField("unit_price", FloatType()),
            StructField("quantity", IntegerType())
        ])
    )),
    StructField("timestamp", TimestampType()),
    StructField("type", StringType()),
])
```

4. **Parsed Json data stream and created data frame with required columns.**

```
# casting data value as string and aliasing
df = lines.select(from_json(col("value").cast("string"), myschema).alias("data")).select("data.*")
df1 = df.select(col("type"), col("country"), col("invoice_no"), col("timestamp"), explode(col("items")))
df2 = df1.select("type", "country", "invoice_no", "timestamp", "col.SKU", "col.title", "col.unit_price", "col.quantity")
```

5.Defined functions for is_order, is_return, total_order_cost and converted them to UDF

```
# defining first function is_order if its a order
def is_order(x):
    if x == "ORDER":
        return (1)
    else:
        return (0)

# defining second function is_return if its a return
def is_return(x):
    if x == "RETURN":
        return (1)
    else:
        return (0)

# defining third function total_cost for  for total order cost
def total_cost(x, y, z):
    if x == "ORDER":
        return (y * z)
    else:
        return ((y * z) * (-1))

# converting pysaprk function to spark UDF in respective datatypes
Total_order_cost = udf(total_cost, FloatType())
Is_order = udf(is_order, IntegerType())
Is_return = udf(is_return, IntegerType())
```

6.Added is_order,is_return,total_cost to the data frame.

```
# Adding total cost, Is_retrun flag,Is_order flag to dataframe df2
df2 = df2.withColumn("total_cost", Total_order_cost(df2.type, df2.unit_price, df2.quantity))
df2 = df2.withColumn("is_order", Is_order(df2.type))
df2 = df2.withColumn("is_return", Is_return(df2.type))
```

7.Defined all the required KPIs

```
# time based KPI
df4 = df2.select("invoice_no", "timestamp", "total_cost", "quantity", "is_order", "is_return")
Final_time = df4.withWatermark("timestamp", "10 minutes") \
    .groupby(window("timestamp", "1 minute")) \
    .agg(sum("total_cost").alias("Total_sales_vol"),
         F.approx_count_distinct("invoice_no").alias("OPM"),
         sum("is_order").alias("total_order"),
         sum("is_return").alias("total_return"),
         sum("quantity").alias("total_items")
    )

# KPI for rate of return
Final_time = Final_time.withColumn(
    "rate_of_return",
    Final_time.total_return / (Final_time.total_order + Final_time.total_return))

# KPI for average transaction size
Final_time = Final_time.withColumn(
    "Avg_trans_size",
    Final_time.Total_sales_vol / (Final_time.total_order + Final_time.total_return))

Final_time = Final_time.select(
    "window", "OPM", "Total_sales_vol",
    "Avg_trans_size", "rate_of_return")

## Time and Country Based KPIs
df5 = df2.select("country", "invoice_no", "timestamp", "total_cost", "quantity", "is_order", "is_return")
Final_KPI = df5\
    .withWatermark("timestamp", "10 minutes")\
    .groupby(window("timestamp", "1 minute"), "country")\
    .agg(
        sum("total_cost").alias("Total_sales_vol"),
        F.approx_count_distinct("invoice_no").alias("OPM"),
        sum("invoice_no").alias("sum_invoice"),
        sum("is_order").alias("total_Order"),
        sum("is_return").alias("total_return"),
        sum("quantity").alias("total_items")
    )

# KPI for rate of return
Final_KPI = Final_KPI.withColumn(
    "rate_of_return",
    Final_KPI.total_return / (Final_KPI.total_Order + Final_KPI.total_return))
```

8. Written the intermediary batch dataset to the console.

```
# printing output on console
query1 = input_table.select(
     "invoice_no", "country", "timestamp", "total_cost",
     "total_items", "is_order", "is_return") \
     .writeStream \
     .outputMode("append") \
     .format("console") \
     .option("truncate", "false") \
     .trigger(processingTime="1 minute") \
     .start()
```

9.Saved the time based and time-country based KPIs in HDFS as JSON files.

```
# Saving Time based KPI as Json in HDFS


query2 = Final_time.writeStream \
    .outputMode("Append") \
    .format("json") \
    .option("format", "append") \
    .option("truncate", "false") \
    .option("path", "time_KPI") \
    .option("checkpointLocation", "time_KPI_json") \
    .trigger(processingTime="1 minute") \
    .start()

# Saving time and country based KPI as Json in HDFS
query3 = Final_country_time.writeStream \
    .outputMode("Append") \
    .format("json") \
    .option("format", "append") \
    .option("truncate", "false") \
    .option("path", "time_country_KPI") \
    .option("checkpointLocation", "time_country_KPI_json") \
    .trigger(processingTime="1 minute") \
    .start()
```

10.Exported the Spark-Kafka version and ran the spark submit command.

```
export SPARK_KAFKA_VERSION=0.10

spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2.11:2.4.5 spark-streaming.py >Console_Output.txt
```

11.Downloaded Time and Country based KPI files into local computer.

```
scp  -r -i ~/Desktop/RHEL_MAC.pem hadoop@ec2-3-92-53-53.compute-1.amazonaws.com:/user/hadoop/time_KPI/  ~/Desktop/time_KPI

scp -r  -i  ~/Desktop/RHEL_MAC.pem hadoop@ec2-3-92-53-53.compute-1.amazonaws.com:/home/hadoop/time_country_KPI
~/Desktop/time_country_KPI_json
```